TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Tub de Bombolles Autònom/SHX

TITULACIÓ: Grau en Enginyeria de Sistemes de Telecomunicació

AUTOR: Albert Martret Torrent

DIRECTOR: Jaime Oscar Casas Piedrafita

DATA: 2 de Setembre del 2015

Resum

Aquest projecte parteix del principi d'educabilitat que diu que no existeix cap nen o cap persona sense la possibilitat de ser educada i d'aprendre, amb més o menys mesura. Per tant, s'estarà en condicions de treballar la comunicació convençuts que els canvis sempre són possibles.

A la nostra societat afortunadament no hi ha una majoria de gent discapacitada. Tot i així, tenim un volum creixent de gent discapacitada que ha de poder accedir a les mateixes tecnologies que la resta.

Hi ha molts tipus de discapacitats i cada una d'elles necessita la seva pròpia aplicació personalitzada. Hi ha empreses dedicades a oferir aquest tipus d'ajuda, adaptar les seves llars i objectes quotidians per la seva fàcil utilització de les seves capacitats.

Al no haver excessiva gent discapacitada, l'adaptació personalitzada de cada cas fa que en sí el producte sigui excessivament car i de difícil accés per a la majoria de la gent.

Aquest projecte té com a objectiu ajudar a desenvolupar les capacitats cognitives visuals de la gent amb necessitats educatives especials, més concretament els que tenen deficiències multi sensorials, que els hi costa tant transmetre el seu coneixement com rebre'l de l'exterior. És el cas dels infants que pateixen paràlisi cerebral, autisme, síndrome d'Asperger, síndrome de Down, entre d'altres, per a que adquireixin els coneixements i els interioritzin més fàcilment.

Per a comunicació, s'entén, tant el llenguatge oral, com llenguatge de signes i altres formes de comunicació no verbal. Un cop inclosa la comunicació no verbal, s'inclouen altres medis i formats augmentatius o alternatius de comunicació, es pot afirmar que per a molts alumnes, la seva forma d'interaccionar amb l'entorn preferent serà la comunicació multi sensorial.

El prototip dissenyat en aquest projecte es podria portar, en un futur, a les sales multisensorials que son el resultat de la recerca d'un entorn predictible, accessible, un entorn que possibiliti als alumnes la seva participació, autonomia, independència i la seva capacitat d'elecció. Un entorn que els permeti, a través de la pròpia experiència sensorial, establir una comunicació amb l'entorn.
Si al disseny d'aquest entorn se li suma la presència d'un adult significatiu que cregui amb la persona, amb independència de la seva discapacitat, donarà com a resultat un entorn afavoridor de l'apertura comunicativa a través dels sentits, en definitiva un entorn afavoridor de la comunicació multi sensorial.

Title: Bubbles Autonomous/SHX Tube

Degree: Telecommunication Systems

Author: Albert Martret Torrent

Director: Jaime Oscar Casas Piedrafita

Date: September 2nd of 2015

Overview

This project is based on the principle of educability which says that every child and every person has the possibility of being educated and learning, with greater or lesser extent. So you will be able to work the media convinced that changes are always possible.

Fortunately, in our society there is a minority of disabled people. However, we have a growing number of disabled people who has to be able to access the same technologies as the rest.

There are many types of disabilities and each one needs its own custom application. There are companies dedicated to offer this kind of help to adapt their homes and everyday objects to its easy use of their abilities.

In the absence of excessive disabled people, the custom adaptation for each case makes itself the product too expensive and inaccessible for the most people.

This project aims to help to develop the visual cognitive abilities of people with special educational needs, specifically those with multi-sensory impairments, who are difficult for them to share and receive knowledge and meet him outside. This is the case of children suffering from cerebral palsy, autism, Asperger syndrome, Down syndrome, among others, to acquire the knowledge and internalize easily.

With communication it is referred to spoken language and sign language and other forms of nonverbal communication. After including nonverbal communication, there are also included other media and augmentative formats or alternative communication, we can say that for many students, the way they interact with the environment will be the referent multi-sensory communication.

The prototype designed in this project could provide, in the future, multisensory rooms which are the result of the search for a predictable environment, accessible environment that enables students to their participation, autonomy, independence and ability election. An environment that allows them, through sensory experience, establish communication with the environment.
If the design of this environment is added to the presence of a significant adult who believes in the person, regardless of their disability, will result in an environment that fosters of openness communication through the senses, ultimately, an enabling environment for multi-sensory communication.

ÍNDEX

INTRODUCCIÓ

“En lo sucesivo, ningún niño debe ser considerado ineducable...
La educación es un bien al que todos tienen derecho [...] Los fines de la educación son los mismos para todos, independientemente de las ventajas o desventajas de los diferentes niños. Estos fines son, primero, aumentar el conocimiento que el niño tiene del mundo en que vive, al igual que su comprensión imaginativa tanto de las posibilidades de ese mundo como de sus propias responsabilidades en él; y, segundo, proporcionarle toda la independencia y auto eficiencia de que sea capaz, enseñándole con este fin lo necesario para que encuentre un trabajo y esté en disposición de controlar y dirigir su propia vida. Evidentemente, los niños encuentran diferentes obstáculos en su camino hacia ese doble fin; para algunos, incluso los obstáculos son tan enormes que la distancia que recorrerán no será muy larga. Sin embargo, en ellos cualquier progreso es significativo” (Warnock, 1990, 12-13). [1]

L’educabilitat no es un privilegi limitat a una determinada població, és pel contrari, un tret distintiu de la espècie humana. No existeix cap persona sense aquesta possibilitat, mentre hi hagi vida, hi haurà canvis. Tampoc existeixen prerequisits ni barreres si l’adult posa tota la seva dedicació amb la comunicació, sobte tot si aposta decididament pel nen.

Tenim un mercat creixent tecnològicament de forma exponencial. Cada vegada l’ésser humà té més facilitats al seu dia a dia gràcies a la tecnologia i pot arribar a fer coses que abans no podia fer, de manera molt més senzilla, fins i tot automatitzar certes tasques dins de la seva pròpia llar.
Bons exemples d’això serien encendre la calefacció o l’aire condicionat quan la temperatura passa d’un cert llindar, baixar persianes quan es detecta que el nivell de llum baixa o encendre llums, regar les plantes quan es detecta que la sorra està seca, etc.

Cada vegada més gent discapacitada vol accedir a aquestes tecnologies ja que poden ajudar-los a ser més autònoms.
Encara que, les persones discapacitades són una minoria de la població mundial, no treu que tinguin el mateix dret que tothom a tenir una vida digne i còmoda, com la resta.

Cada persona discapacitada té unes necessitats que no tenen per què ser les mateixes que la de la resta, per tant, es crea una demanda molt personalitzada que donarà lloc a un futur producte que, pels temps que corren, serà poc econòmic.

Precisament serà en aquest problema amb el que se centrarà aquest projecte.
Es vol dissenyar un prototip d’un servei d’ajuda per a millorar l’aprenentatge cognitiu visual dels infants amb la discriminació dels colors i el premi del reforçament positiu adaptat als infants amb necessitats educatives especials

basat amb un tub de bombolles de colors i que, a més, tingui un cost accessible per a qui ho demana.

Per a fer-ho possible, s'utilitzarà software que es pot descarregar gratuïtament i s'utilitzaran els materials de manera eficient, buscant els més econòmics, ja que, al ser un prototip es buscarà el material just per a que funcioni.

Durant el document s'explicarà quines necessitats exactament es volen cobrir, què s'ha pensat per dur-ho a terme i quins materials s'han comprat per a que sigui més econòmic tenint la mateixa funcionalitat i sent de qualitat.

CAPÍTOL 1. NECESSITAT I OBJECTIUS

1.1. Situació actual

Actualment, tot tipus d'ensenyament està basant en el reforçament positiu. És un premi per la resolució d'un objectiu concret, premiant-lo positivament. Aquest premi està adaptat als infants ja que necessiten un estímul que ells valorin i ajudin a millorar.

Per exemple, no serà el mateix premi a un nadó que està aprenent a caminar que a un nen que aprèn a llegir i escriure que a un alumne de Primària o la ESO.
En els primers casos, el reforçament positiu serà més sentimental, amb bones vibracions i sensibilitat. En els últims casos ja seria posar una bona nota contra millor sigui l'actitud de l'infant.

Hi ha infants, pel contrari, que els estímuls a aprendre en els últims casos, no els valoren ni els entenen, així doncs, s'hauria d'aplicar un altre tipus d'estímul adaptat a l'infant i personalitzat, per exemple, relacionat amb el que ell més li agradi.

A tota escola hi ha infants amb aquestes necessitats on el currículum se'ls hi adapta.

1.2. Què son les funcions cognitives?

Són aquells processos mentals que ens permeten raonar, pensar i resoldre problemes. Inclouen habilitats com comprendre i utilitzar el llenguatge, mantenir i dividir l'atenció, aprendre i recordar informació nova, reconèixer objectes i classificar-los a l'espai, etc.

Aquesta sèrie d'habilitats varia de forma natural entre les persones fent que cadascú tingui els seus punts dèbils i els seus punts forts.
El funcionament cognitiu es considera normal si cadascú a casa pot resoldre correctament les activitats de la vida diària.

Els trastorns cognitius més directes inclouen la amnèsia, la demència i el delírium. Uns altres, inclouen el trastorn d'ansietat com les fòbies, pànics, etc.
Els trastorns de l'estat d'ànim també son trastorns cognitius com la depressió i el trastorn bipolar.
També està inclòs dins del trastorn cognitiu els trastorns psicòtics, com l'esquizofrènia i els trastorns delirants classificats com trastorns mentals cognitius.[2]

1.2.1. Funcionament cognitiu a la síndrome d'Asperger

Veiem un exemple de les característiques del funcionament cognitiu a la síndrome d'Asperger, és un derivat dels trastorns de l'espectre autista i es caracteritza per una limitació significativa de les capacitats de relació i comportament social.

Les persones amb síndrome d'Asperger tenen un nivell d'intel·ligència normal o superior a la mitja. Tot i així, presenten dèficits cognitius i un estil d'aprenentatge peculiar que afecten a l'accés del currículum acadèmic (requerint aprenentatge específic) i al seu funcionament quotidià.
Entre les dificultats cognitives que podrien obstaculitzar la seva adaptació trobem, entre d'altres:

- **Dèficit de cognició social:** És un tret característic que comparteix amb altres trastorns d'espectre autista, el dèficit en el desenvolupament de la teoria de la ment. Es manifesta a les seves dificultats per entendre i atribuir estats mentals als altres i a un mateix com desitjos, creences o intencions.
- **Dèficit en habilitats d'organització i planificació:** Mostren dificultat per formar una representació interna de l'objectiu final de la tasca a realitzar. Tindran dificultats per saber col·locar-se als espais oberts o com anar d'un lloc a un altre, tindre previst el material per realitzar una tasca o finalitzar la tasca dins el temps establert.
- **Rigidesa mental:** Manifesten dificultats per a contemplar diferents alternatives de solució d'un problema i analitzar la informació des de diferents punts de vista. Tenen dificultats per desplaçar de forma flexible el focus de la seva atenció. Tendeixen a adherir-se de forma rígida a les seves opinions.
- **Falta de motivació per l'aprenentatge:** Una característica molt típica de la síndrome d'Asperger es un interès molt intens en un tema molt específic sobre el que acumula gran quantitat d'informació. Per altra banda, pot manifestar molt poca motivació per a tota la resta. L'absència de motius competitius, així com la indiferència de l'infant cap al reforç social, dificulten molt l'ensenyament dels continguts pels que el nen no mostra cap interès personal.

Les fonts de motivació social no solen tenir gaire efecte. Podria ser que no estiguessin tan interessats en agradar als altres o que no s'identifiquessin amb els adults que admiren. Les dificultats amb la capacitat d'anticipar i en donar sentit a l'acció pròpia pot convertir en ineficaços els incentius a llarg termini i les consideracions pel futur. [3]

1.2.2. Funcionament cognitiu a la Paràlisi Cerebral

En el cas de la paràlisi cerebral, que serà majoritàriament amb el tipus de malaltia que se centra el projecte, s'utilitza un mètode d'estimulació dels

diferents sentits per aconseguir millorar el funcionament cognitiu ja que els malalts tenen un aprenentatge molt limitat.
L'estimulació multi sensorial es un instrument utilitzat amb l'objectiu de millorar les condicions de vida de les persones amb discapacitat. Per fer-ho, es recorre a estratègies que treballen les capacitats més bàsiques del ser humà: les sensacions, la percepció i la integració sensorial.
Amb aquests tractaments no es pretén curar a aquests discapacitats, però sí fer que gaudeixin, millorin les seves capacitats cognitives i de relació i que es trobin millor.

Als infants, es busca reforçar el seu desenvolupament afavorint la integració de la informació sensorial que reben, ajudant en els seus aprenentatges i la seva relació amb l'entorn. En aquest cas seria amb l'ajuda del motor de bombolles, els LEDs i el so.
Es treballen els sentits en un ambient d'estímuls controlats, on es faciliten la exploració, el descobriment i gaudir de diferents experiències sensorials arribant a experimentar sensacions intenses amb la possibilitat d'expressar emocions contingudes. Es busca un despertar sensorial a traves de la pròpia experiència sensorial. [4]

1.3. Motivació i Objectiu del Projecte

L'objectiu d'aquest projecte es millorar notablement aquest reforçament positiu als alumnes amb necessitats educatives especials, permetent que l'alumne s'endinsi en un món de llums de colors amb un tub de bombolles que van canviant de color per donar una major sensació de riquesa lumínica i perceptiva i que es pugui interactuar amb aquest tub de bombolles amb una amplia gama de possibilitats.

Després d'haver realitzat l'exercici positivament, aquest dispositiu pot servir per a endinsar-se en un món específic preferit, provocant que l'infant tingui ganes de tornar-hi o prendre's-ho com un joc.

A la vegada també es poden treballar els colors primaris, estimulant visualment i ajudant a introduir-se en l'aprenentatge. Ja sigui amb l'elecció correcta d'un color o identificant-los.
També es podran treballar diferents sentits a la vegada, com la música i el tacte, gràcies al tub de bombolles i els colors i l'acció reacció dels mateixos en el cas que polsin algun botó.

A les sales multi sensorials, es poden utilitzar aquests materials com a instrument d'avaluació de l'infant, avaluant la resposta a diferents estímuls, o com a instrument d'intervenció per ajudar a l'infant a recolzament positiu i rehabilitador i per la recuperació d'hàbits, conductes, comportaments, etc.
Així mateix es poden utilitzar aquestes sales com a instrument de comunicació multi sensorial, que permet als alumnes greument afectats poder-se manifestar, relacionar-se amb el món i donar-li un mínim d'independència a la vegada que serveix com a canal d'informació i coneixement de l'infant.

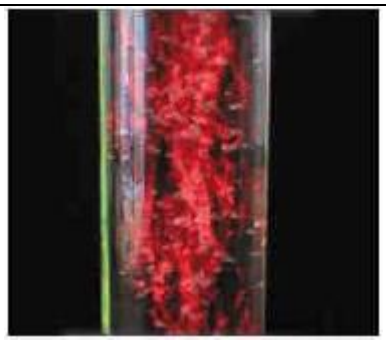
CAPÍTOL 2. ARQUITECTURA DEL HARDWARE

2.1. Funcions principals del projecte

Per poder portar a terme el projecte es va contactar amb una empresa que treballa amb gent discapacitada i que té estudis de primera mà de les seves necessitats: BJ Adaptaciones[5]. Ella ens va donar uns objectius a assolir que podrien ajudar-los en un futur a crear un producte que podria tenir interès comercial.

El que es volia fer era agafar un tub de bombolles que hi ha al mercat avui en dia, tipus Spacekraft, per exemple, adaptar-lo al seu sistema i millorar les condicions d'autonomia, preu i funcionalitats.

Al mercat hi ha diferents tubs amb diferents objectius finals i característiques[6]:

NOM	IMATGE	DESCRIPCIÓ
Columna Bombolles Activa (al voltant de 1000€)		✓ Tenen unes bombolles que canvien lentament de color. ✓ Aquest canvi constant pot interrompre's en qualsevol moment amb un interruptor. ✓ Té diverses mesures de diàmetre i longitud. ✓ Gràcies al seu canvi de color, vibració i bombolles, proporciona uns sorprenents estímuls a l'usuari. ✓ Estimulació de vista i tacte gràcies al canvi de color i la vibració de la paret de la columna.

Columna Bombolles Sensible al So (al voltant de 1000€)		✓ Responen al so canviant de color en un rang de 16 colors. ✓ La sensibilitat del so es pot ajustar ✓ Té diferents mesures de diàmetre i longitud.
Columna Bombolles Interactiva (de 800€ a 2000€)		✓ Incorpora un polsador amb 4 botons que, polsant-los de forma individual o conjunta pot produir una gama de 16 colors. ✓ Té diferents mesures de diàmetre i longitud. ✓ Gràcies al seu canvi de color, vibració i bombolles, proporciona sorprenents estímuls a l'usuari. ✓ Estimulació de vista i tacte gràcies al canvi de color de les bombolles i la vibració de la paret de la columna.

Tabla 1: Tipus de Tubs de Bombolles

A partir de l'estudi de les característiques i mancances d'aquests sistemes comercials i de les necessitats a l'educació dels potencials usuaris amb discapacitat es van definir les característiques que havia de tenir el disseny a realitzar. Aquestes queden exposades en el següent apartat.

2.1.1. Característiques

Aquest apartat parla sobretot de les característiques del hardware que, a partir dels estudis amb nens amb discapacitat i de l'estat de la tècnica, es van decidir que tingués el projecte. Aquestes característiques queden enumerades a continuació i la majoria superen l'estat de l'art amb més botons i diferents interfícies d'usuari.

- El prototip haurà de tenir una sortida per a el control dels LEDs RGB amb nivell d'il·luminació equivalent als dispositius existents, com el tub de bombolles o les fibres òptiques, similars als ja existents.

- Haurà de tenir 9 entrades de commutadors i una entrada amb connector tipus DIN per a la botonera amb sortida de 12V per a la retro il·luminació del teclat, cosa que marca novetat al mercat ja que no existeix cap teclat retro il·luminat ni amb tants botons.
- Haurà de ser compatible amb el Sistema de BJ de Radiofreqüència (RF). El comandament a distància, tampoc existent en cap model anterior perquè és exclusiu de BJ.
- Tindrà un connector que funcionarà amb el prototip d'il·luminació DMX, que utilitzen les discoteques i els escenaris. Els models fins ara trobats recreen llum de diferents colors però el mètode és automàtic preestablert o s'indica el color mitjançant botons, en canvi amb el DMX es podrà monitoritzar i controlar des d'un ordinador, si es vol.
- Haurà de tindre una entrada de micròfon que detecti la potència amb la que es parla per ell, similar al tub sensible al so.
- Estarà compost per un relé de sortida per controlar el motor de bombolles.
- També com a novetat, ja comentada anteriorment, a més, es va decidir que el tub tingui un selector que es pot escollir entre RF i canal DMX.
- Finalment, s'especifica que el prototip estigui el més optimitzat possible per a la possibilitat que funcionés amb bateries.

2.1.2. Modes de Funcionament

El prototip constarà de 2 modes de funcionament que s'escollirà mitjançant un selector:

- *Mode Autònom*

Dintre del mode autònom, escollit prèviament amb el selector, trobarem diverses entrades tant per commutadors com pels botons del comandament a distància. Cada botó realitzarà una tasca diferent:

1. Activarà/Desactivarà el relé del motor de bombolles
2. Els LEDs seran vermells
3. Els LEDs seran verds
4. Els LEDs seran blaus
5. Els LEDs seran grocs
6. Els LEDs seran color magenta
7. S'apagaran tots els LEDs
8. Es passarà al Mode Automàtic, que estarà posat per defecte al arrancar.
9. Mode Micròfon, que anirà canviant de color cada cop que la senyal del micròfon superi un cert nivell.

Al encendre el dispositiu, la roda de colors anirà canviant suaument, tal i com fan els tubs Spacekraft, que correspondrà al botó 8.

- *Mode Sistema SHX (DMX)*

El Sistema SHX és el que conjunt d'elements interconnectats que creen una zona sensorial controlable per l'usuari en tot moment. Aquest mode establirà el color que han de tindre els LEDs RGB mitjanant el protocol DMX. Només s'utilitzarà la entrada del commutador del motor de bombolles.

2.2. Arquitectura del Sistema

Després d'analitzar les necessitats i funcionalitats establertes del projecte, va quedar una estructura que es podria resumir en el següent diagrama de blocs.

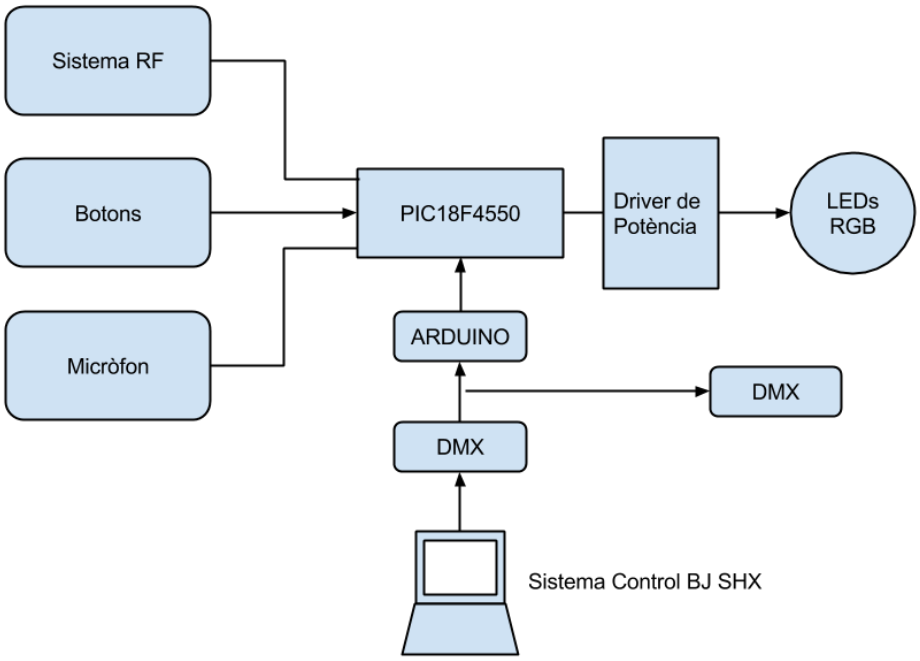


Figura 1: Estructura del Sistema

Com es pot veure a la imatge, el microcontrolador es el cervell del sistema. Tindrà múltiples entrades que li demanaran les diferents funcionalitats però només una sortida (LED RGB), la qual està dissenyada per respondre a totes les demandes.
A aquesta sortida de LED hi anirà un driver de potència que possibilitarà el control de la il·luminació amb la Modulació per Ample de Polsos (PWM) i la futura ampliació de la quantitat de LEDs que es desitgin.
El mètode PWM fa, de manera digital, que les llums dels LEDs tinguin un rang d'il·luminació visible pràcticament analògic. Aquest mètode juga amb el filtre passa-baixos, que forma tant les capacitats dels LEDs com l'ampla de banda de visió de l'ull humà. Contra més intensitat pampalluguegi, més alta serà la mitja a la que el LED està encès i, per tant, més alta veurem la il·luminació del LED.

Tot i que el microcontrolador principal s'encarrega de processar totes les senyals, existeix un segon microcontrolador especialitzat en el protocol DMX, on, en aquest cas, controla només tres canals d'un mateix univers.
Es va trobar que era millor tenir 2 microcontroladors que treballessin en paral·lel, que no un que s'encarregués de tot ja que la càrrega del protocol DMX es molt alta i ens interessa una resposta ràpida per part del sistema. A més, aquest segon microcontrolador, té la capacitat de suportar una ampliació de canals i universos en el cas que es desitgi, sense alterar el comportament del sistema.
En el projecte no es tractarà la etapa de potència, sinó que està centrat en la resta, les comunicacions.

En els següents capítols es farà una breu descripció de cada component del sistema i quin model s'ha escollit com a més adient per a dissenyar el prototip.

CAPÍTOL 3. DISSENY DEL HARDWARE

3.1. Nucli de Control del Tub de Bombolles

3.1.1. LEDs RGB

3.1.1.1. Què és un LED?

Un LED (de l'acrònim anglès *LED*, *light-emitting díode:* 'díode emissor de llum') es un component optoelectrònic passiu i, més concretament, un díode que emet llum.

Els LEDs s'utilitzen com indicadors en molts dispositius i en il·luminació. Els primers LEDs emetien llum vermella de baixa intensitat, però els dispositius actuals emeten llum d'alta lluentor a l'espectre infraroig, visible i ultra violeta.
Degut a la seva capacitat d'operació a altes freqüències, són també útils en tecnologies avançades de comunicacions i control. Els LEDs infrarojos també s'utilitzen en comandaments a distància de molts productes comercials, incloent els equips d'àudio i vídeo.[7]

3.1.1.2. LED RGB

Un LED RGB es aquell LED que té dins seu els 3 LEDs colors primaris. El vermell, el verd i el blau. Segons com es connectin i la intensitat de cada color, es pot tindre una gamma complerta de tots els colors de l'arc iris.
Aquest tipus de díode es el que s'ha escollit com a dispositiu de sortida del prototip ja que pot satisfer totes les demandes de les diferents entrades.
Per aquest projecte només es treballarà amb un sol LED però per a l'aplicació final es posaran tants LEDs en paral·lel com faci falta amb el driver de potència per controlar la intensitat ta i com s'ha explicat anteriorment..

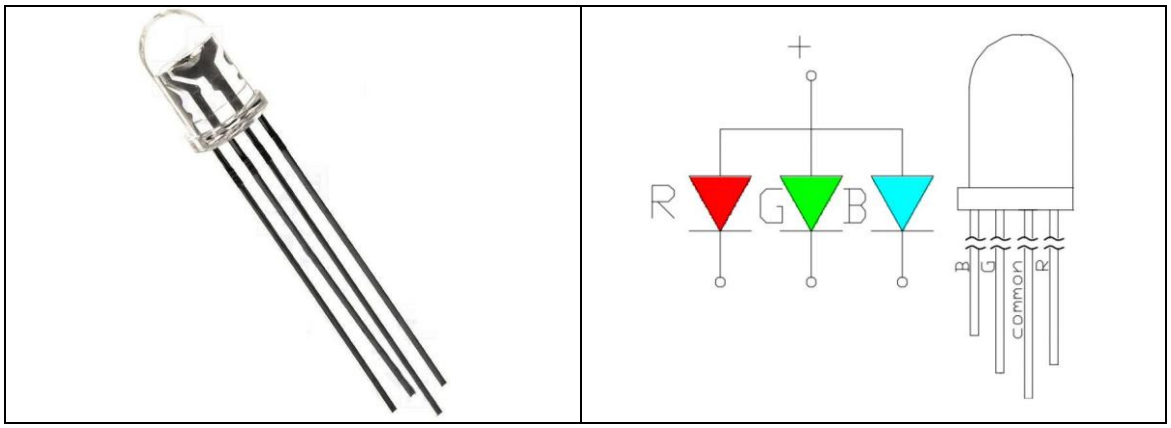


Tabla 2: Un LED RGB i el seu Esquema

3.1.2. Microcontrolador

3.1.2.1. Què és un Microcontrolador?

Un microcontrolador és un microprocessador especialitzat a controlar equips electrònics, i inclou en un sol xip les tres unitats funcionals d'un ordinador: una CPU, memòria i unitats d'E/S (Entrada/Sortida), és a dir, es tracta d'un computador complet en un sol circuit integrat.
Emfatitza l'alta integració, en contrast amb un microprocessador que només conté una CPU. A més de les típiques operacions lògiques i aritmètiques d'un microprocessador de propòsit general, un microcontrolador integra elements addicionals com memòria de lectura-escriptura per a dades, memòria de només lectura per emmagatzemar el programa, memòria flaix per emmagatzemament permanent perifèrics, i interfícies d'entrada sortida (Ports, I2C, SPI...).

Consumeixen poca energia (miliwatts o fins i tot microwatts), i en general mantenen la funcionalitat mentre esperen un esdeveniment com prémer un botó o una interrupció. El consum d'energia en estat inactiu (rellotge de la CPU i perifèrics desactivats) pot ser de només nanowatts, sent ideals per aplicacions de baix consum i bateria duradora. Encara que n'hi ha de la mida d'un segell de correus, el normal és que siguin encara més petits, ja que, lògicament, formen part del dispositiu que controlen.[8]

- *Què és un PIC?*

Son una família de microcontroladors fabricats per Microchip Technology Inc. I derivats del PIC1650, originalment desenvolupat per la divisió de microelectrònica de General Instrument.
El nom actual no és un acrònim. En realitat, el nom complert es PICmicro, tot i que generalment s'utilitza com **P**eripheral **I**nterface **C**ontroller (Controlador d'Interfície Perifèrica)

- *Què és un Arduino?*

És una placa de circuit imprès simple basada en el microcontrolador de codi obert provinent de la plataforma de codi obert Wiring, amb l'objectiu de fer més simple i accessible el disseny de circuits electrònics amb microcontroladors.

3.1.2.2. Elecció del Microcontrolador

BJ Adaptaciones va oferir diversos microcontroladors tipus PIC amb els que treballen ells per poder realitzar el projecte.
Per altra banda, es va veure necessària la intervenció d'un segon microcontrolador tipus Arduino, del que es parla més endavant, dedicat a tasques específiques per poder treballar en paral·lel els ambdós microcontroladors i no tenir retards a les respostes del sistema.

- *Elecció del PIC*

Es va escollir, principalment, el PIC que més ports tenia, per poder tindre més llibertat de disseny sense haver de modificar el PIC per a futures ampliacions. També entre els PICs que tenien més ports es va escollir el que treballava amb un convertidor Analògic-Digital (AD) de 10 bits, Memòria Flash i que potencialment més s'ajusta als requeriments del projecte.
La memòria Flash significa que el PIC es reprogramable, per tant, sobreescrivible. En aquest cas la memòria es de 32 Bytes.
Té 5 ports bidireccionals (Entrada/Sortida). Ports A, B, C, D i E
El rang de funcionament del PIC es de 4,2 a 5V i la sensibilitat del port d'entrada analògic està al voltant dels 5 mV.

El model escollit va ser el **PIC18F4550**.

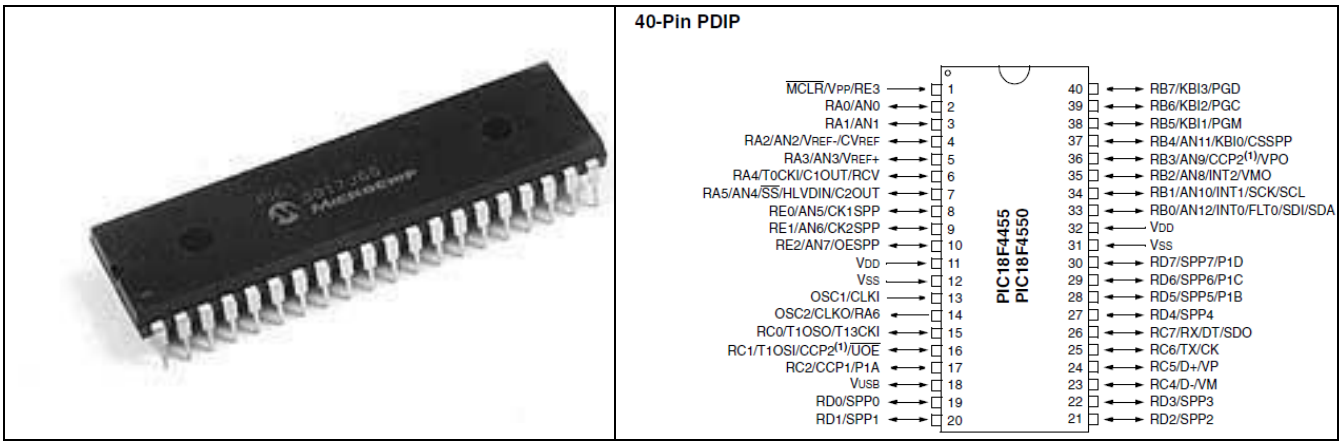


Tabla 3: Imatge esquemàtica i real del PIC

- *Elecció de l'Arduino*

L'elecció de l'Arduino va estar inspirada en un projecte que van fer a la Universitat Politècnica de Pachuca, Mèxic, on expliquen com van fer un controlador digital mitjançant el protocol DMX-512 [9].
D'allà extreuen també la llibreria dels 4 universos i utilitzen una versió de software determinada amb un Arduino UNO, el qual també es va comprar per realitzar aquest projecte.

3.1.3. Micròfon

Hi ha dues maneres d'escollir el micròfon. La digital i la analògica.
En l'estudi previ, es va veure que el micròfon que donava el resultat analògic, s'havia de processar la informació, filtrar-la, passant-la a digital amb un convertidor, i processar la senyal digital per a poder fer el que es demanava.

El micròfon que donava la sortida digital, en canvi, ja incorporava el convertidor analògic digital i es podia introduir el senyal al microprocessador i ja programar el que es desitgés.

En aquest cas, es va escollir el micròfon analògic ja que l'empresa anava a treballar amb el típic micròfon amb connexió de jack habitual, tot el processat de la senyal no caldria fer-lo ja que el PIC ho faria automàticament configurant el port d'entrada com a entrada analògica.
El micròfon escollit va ser el ABM-713-RC que es un micròfon analògic de condensador electret que treballa entre 2 V i 10 V i té una sensibilitat de -42 dB. Aquest micròfon va connectat a un amplificador de senyal LM386N-3 ja que el micròfon utilitzat no té preampliació i així es podia adaptar la sortida del micròfon al marge dinàmic d'entrada del convertidor analògic digital del microcontrolador.

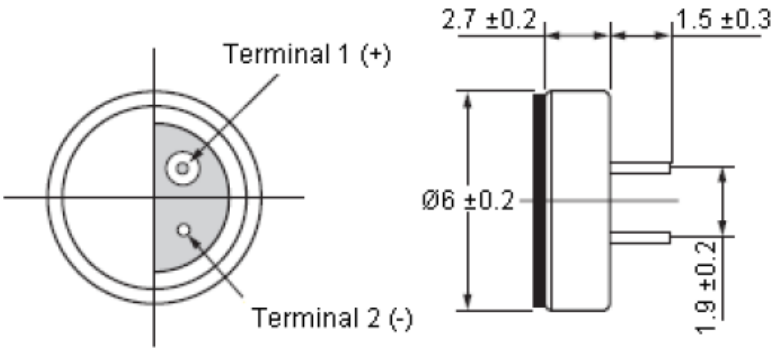
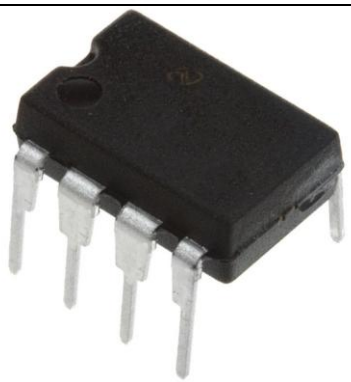
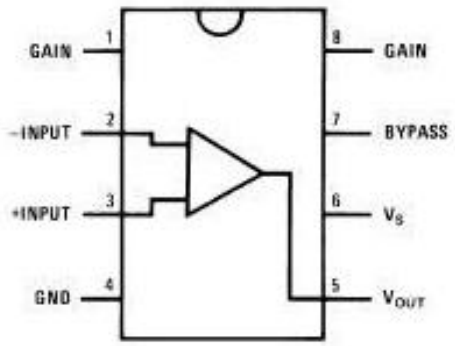
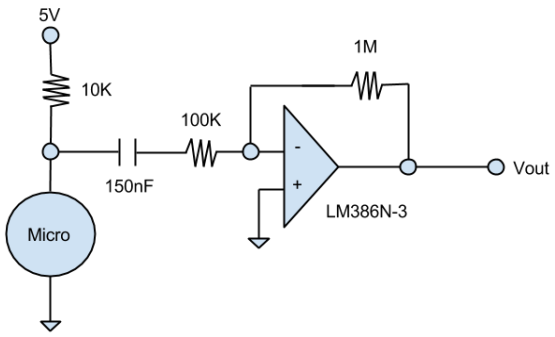
	Dispositiu Físic	Esquema del Dispositiu
Micròfo ABM-713-RC		
Amplificador LM386N-3		
Circuit Micròfon		

Tabla 4: Components del Micròfon i Esquemes

3.1.4. Sistema de Radiofreqüència BJ Live

Amb l'experiència i investigacions prèvies, BJ Adaptaciones va crear un comandament a distància que facilita totes les tasques que es vulguin adaptar per configurar-les dins d'aquest comandament. Aquest sistema, compost pel comandament i la recepció dels canals, va ser facilitat per incloure'l dins el projecte. El comandament a distància està composat per 10 - 12 botons i emet a una freqüència de 434 MHz de la banda lliure ISM. El sistema de recepció té una antena i dóna resposta als impulsos rebuts pel comandament.
Aquests impulsos s'han agafat i s'han posat als diferents pins d'entrada del PIC mitjançant cables, així, serà el mateix polsar amb els botons del comandament a distància, que amb els botons normals.
La influència que pot tindre això a la hora de dissenyar el software es mínima, ja que per a cada funció que realitzarà el botó, farà la mateixa funció polsant el botó de RF corresponent.

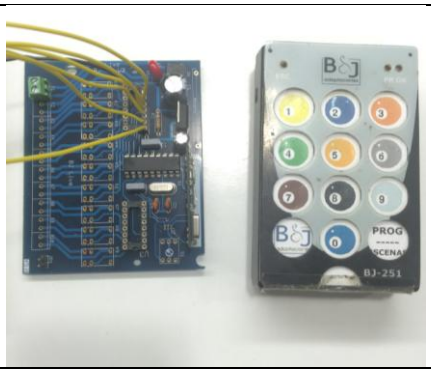
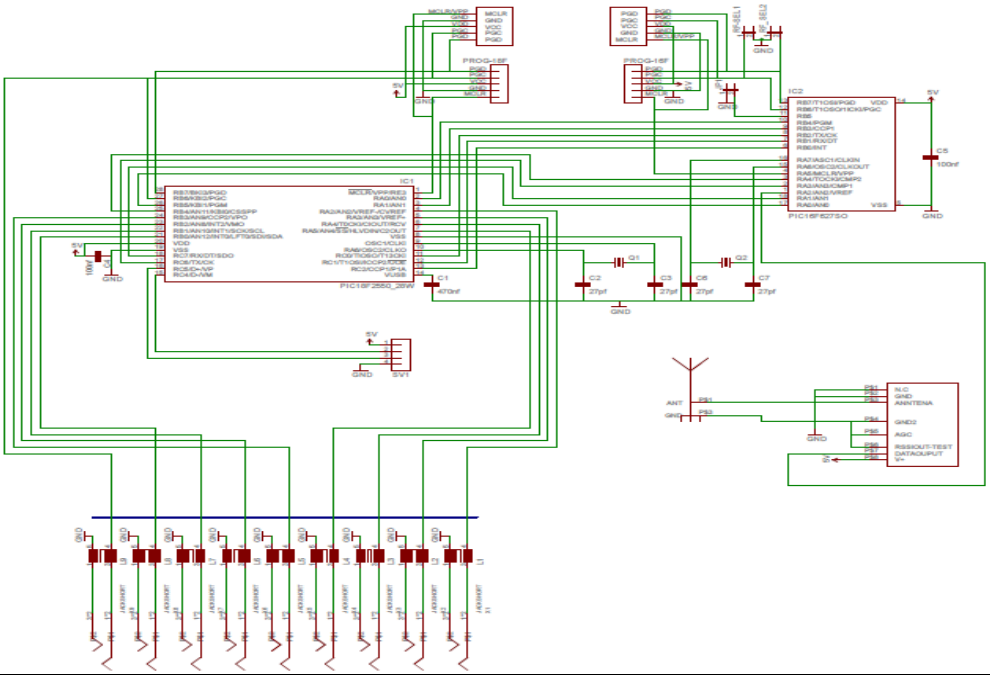
Sistema de RF de BJ	
Esquema del Sistema de RF intern	

Tabla 5: Sistema RF i Esquemàtic

3.2. Estructura (bus) del Sistema DMX

En aquest cas, el cervell que controla aquest sistema, o sigui, el microcontrolador es l'Arduino anteriorment esmentat.

El pes del protocol DMX i la informació que porta pot arribar a ser tan gran que no interessava perdre velocitat de resposta del sistema en futures ampliacions i per això es va optar a utilitzar els 2 microcontroladors (PIC + Arduino), d'aquesta forma, es podran realitzar diferents treballs en paral·lel. L'Arduino bàsicament s'encarrega del processat del senyal DMX que arriba de l'ordinador, el processa i passa la informació al PIC que realitzarà finalment la sortida en format llum de LED.

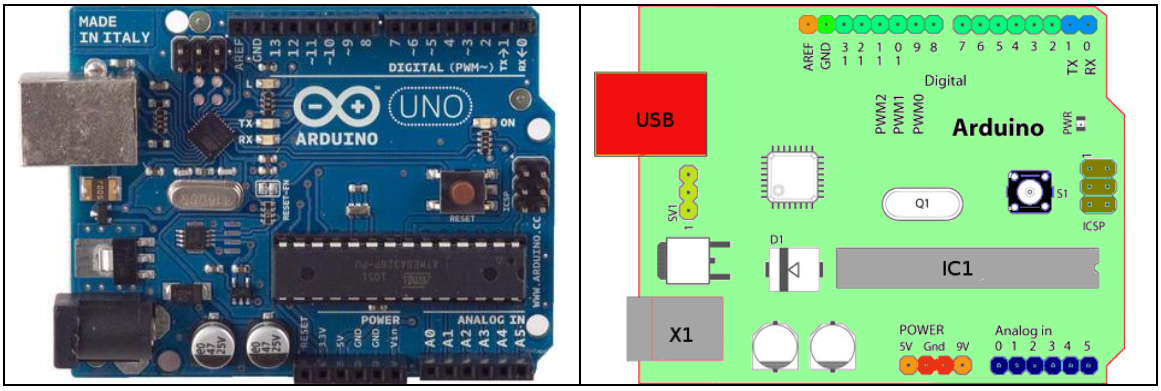


Tabla 6: Imatge i esquema de pins

Ambdós microcontroladors són de mesures relativament petites en comparació amb el sistema de LEDs que s'utilitzarà i el tub de bombolles, així que no s'ha trobat problema a l'hora de decidir utilitzar 2 microcontroladors en contes d'un com estava establert en un inici.

3.2.1. Protocol DMX

Per veure el pes que el protocol DMX aporta al sistema i la velocitat amb el que s'ha de processar les dades, es farà una breu explicació del protocol.

3.2.2. Introducció al DMX

El protocol DMX, ve de l'abreviació de **D**igital **M**ultiple**X**, és un protocol de luminotècnia per al control de la il·luminació d'espectacles i permet la comunicació entre els equips de control de llums i les pròpies fonts de llum.

El protocol apareix com la solució al problema de la incompatibilitat que hi havia entre marques per la utilització de protocols propis, el que obligava a tindre un control del protocol de cada marca de llums amb la que es treballava.

3.2.3. Canals i Valors del DMX

DMX es basa amb la utilització de canals per transmetre ordres de controls als aparells que el suportin. El protocol DMX512 te un límit de 512 canals, cada canal es pot regular des del valor 0 fins el valor 255 associats aquests nivells als nivells d'intensitat lumínica, essent 0 apagat i 255 totalment encès.

3.2.4. Configuració del Sistema DMX

El senyal DMX pot ser entrellaçat entre dispositius a través d'una connexió en cascada. El cable DMX amb el senyal original surt d'un controlador DMX i s'envia al primer dispositiu de l'enllaç DMX. Tots els dispositius amb suport per DMX tenen connectors DMX d'entrada i de sortida. Així doncs, des del connector de sortida del primer dispositiu es connecta un altre cable DMX que es dirigeix al connector d'entrada del següent dispositiu i així successivament. Al final de l'enllaç DMX, es a dir, al connector de sortida de l'últim dispositiu, sempre es recomanable col·locar un "acabador" DMX (DMX terminator) que tanca l'enllaç (resistor amb una carga de 120 ohms) entre els pins 2 i 3 del connector XLR.

3.2.5. Adreça DMX

Es dedueix que el senyal DMX enviat des d'un controlador, conté comandes DMX per a tots els dispositius que hi ha a l'enllaç i que el senyal DMX no te forma de saber a on estan anant aquestes comandes. Per això es necessita la configuració de la adreça DMX a cada dispositiu.
Si tenim 3 dispositius al nostre enllaç, que utilitzen cada un 5 canals DMX, aleshores, l'adreça DMX del primer dispositiu es pot configurar a 1 (de l'1 al 5), la segona al 6 i el tercer a l'11, per exemple.

3.2.6. Configuració del cable DMX

El DMX, doncs, es un protocol de comunicació que utilitza el RS485 com a estàndard físic de comunicació i té una configuració del cablejat específica.

La configuració dels pins de l'1 al 3 a un cable de 3 pins es la mateixa que la dels pins 1 al 3 en un cable de 5 pins.
Un connector de 5 pins (XLR-5) està configurat de la següent manera:

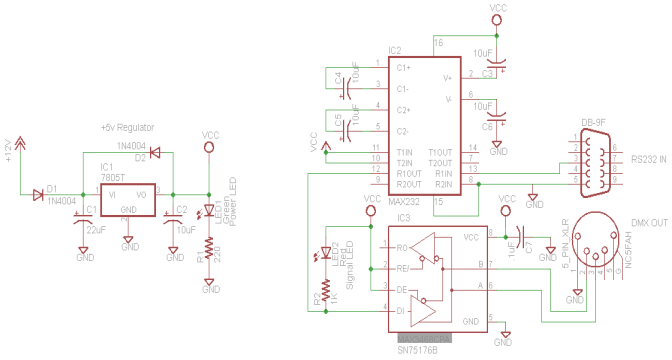
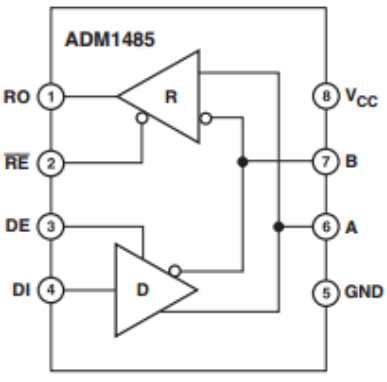
- Pin 1 = senyal de referencia
- Pin 2 = senyal invertida = "-" pol negatiu
- Pin 3 = senyal = "+" pol positiu
- Pin 4 = opcional. (originalment era per tindre feedback dels dispositius i que fos bidireccional)

- Pin 5 = opcional. (igual que al pin 4) [10]

3.2.7. Elecció del Sistema DMX

Per a realitzar les proves amb el prototip, en contes de tindre un sistema DMX original, BJ va facilitar per al projecte un sistema d'il·luminació hardware i un software.
El sistema consisteix en que el software simula un aparell que funciona amb el protocol d'il·luminació i emet un senyal de color per una UART.
Aquesta informació passa al sistema DMX i la converteix a color real amb la sortida i el cablejat corresponent al DMX.
Finalment, aquesta sortida serà la entrada al PIC d'on farà la funció que li correspon amb aquesta senyal.
El PIC només suporta senyals digitals, pel que abans de rebre el senyal del DMX faltarà passar la senyal a diferencial.

Com s'ha comentat anteriorment, el protocol DMX funciona amb el protocol de comunicació RS485.
Per a processar dades d'aquest protocol, es va trobar que ja existia un dispositiu diferencial que canviava la senyal rebuda del DMX a senyal digital quadrada, que pot llegir el PIC per així poder processar-la o en aquest cas, copiar-la i transferir-la sense ser modificada. Aquest dispositiu es el ADM1485 (RS485).

	Dispositiu Físic	Esquema del Dispositiu
Sistema DMX		
ADM1485 RS485		

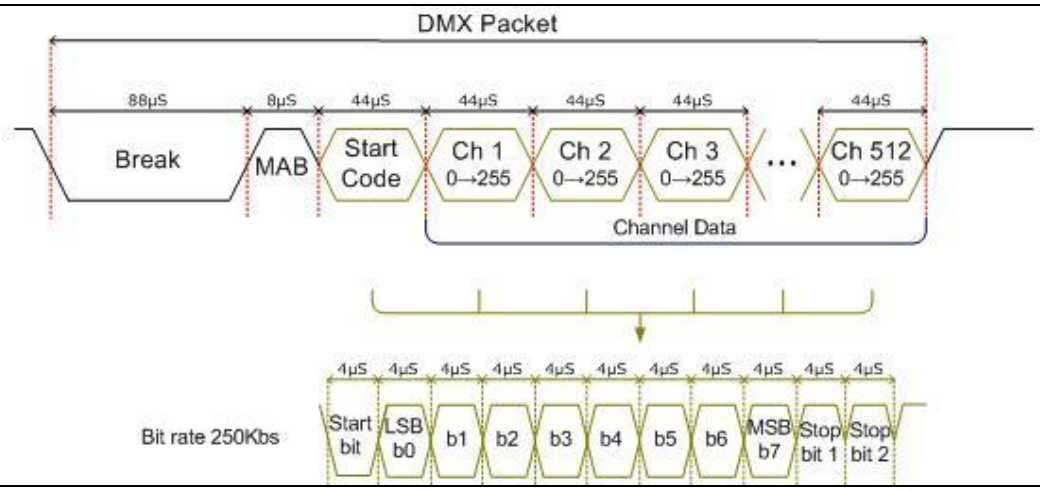
Paquet DMX512	

Tabla 7: Dispositius DMX i Esquemes

La última imatge de la taula mostra la estructura d'un paquet del protocol DMX on comença amb un tram que no hi ha cap transmissió, la marca després del descans (MAB), codi d'inici i finalment, les transmissions dels 512 canals.
Cada canal té una estructura interna de un bit que indica l'inici de la trama, un byte (8 bits) d'informació, més els dos bits finals que son de finalització de la trama.

3.3. Integració Total del Sistema i Conclusió

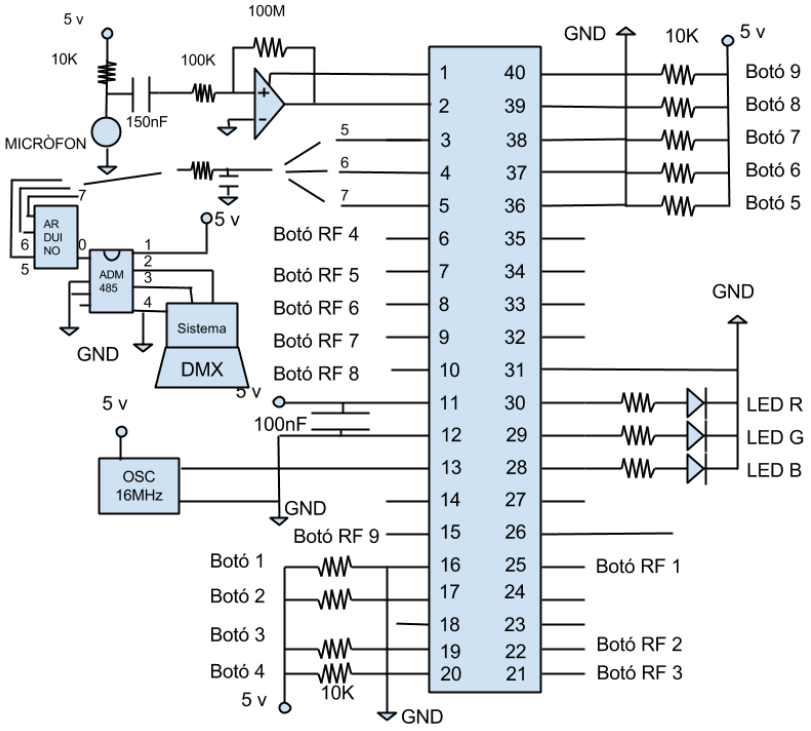


Figura 2: Circuit general

En la figura anterior es mostra una reproducció de l'esquema del circuit utilitzat en el projecte.
En els pins 3, 4 i 5 es realitzen 3 filtres passa baixes, un per cada entrada des de l'Arduino. En la imatge només hi ha representat un filtre.

Com s'ha comentat anteriorment, la proposició d'utilitzar un sol microcontrolador es va descartar poc després de començar el projecte degut a la càrrega de dades que pot suportar el protocol DMX. Utilitzant un segon microcontrolador especialitzat exclusivament amb aquest protocol, es pot donar més marge d'estabilitat i resposta al sistema sense saturar-lo o que vagi excessivament lent.
La diferència d'utilitzar un o dos microcontroladors no afecta a l'aspecte físic, o sigui, no es important que ocupi un espai més gran amb el segon microcontrolador ja que treballem amb una placa de LEDs molt superior en mesura al conjunt dels microcontroladors i ens dona una resposta molt més bona.

CAPÍTOL 4. DISSENY DEL SOFTWARE

4.1. Programar un PIC

En el capítol anterior s'ha explicat què era un PIC i per a què servia a nivell de hardware. Ara s'analitzarà com funciona a nivell de software, que es pot descarregar gratuïtament.
Un PIC, significa Circuit Integrat Programable (**P**rogramable **I**ntegrated **C**ircuit).

Dividim el procés de Programar un PIC en 4 passos:

- Editar
- Compilar
- Gravar
- Provar el programa

El primer pas es escriure el codi amb l'objectiu que el PIC acabi fent el que se li ha demanat. Aquest procés es el d'editar.

Per a compilar el PIC s'utilitza el compilador MPLAB-XC8 (Microchip). Compilador de la casa Microchip, el mateix fabricant dels PIC, concretament pels PIC de 8 bits.
S'ha utilitzat aquest tipus per acord arribat amb BJ.

Gravar o cremar el PIC es el procés que passa d'una versió que entenen els humans a una versió més bàsica a llenguatge màquina on el que ho entén es el propi PIC.

Finalment es testeja el PIC amb el hardware corresponent, en aquest cas els LEDs i s'espera que tingui la resposta desitjada. En el cas que no tingui una resposta correcta, tornarem al primer pas, a 'Editar'.

4.2. Programació Basada en Objectes

L'única limitació que es va posar a la hora de crear i programar el software va ser que el codi s'escrivís amb el llenguatge C i amb assemblador per al protocol DMX.
El funcionament del software a grans termes fa que, depenent del botó que es polsi, el sistema farà una funció o una altra amb la resposta visual corresponent als diferents colors del LED encès.
No només es va programar el codi que identificava la diferència de polsar un botó o un altre sinó que es van configurar els pins d'entrada i sortida, tant si aquesta ha de ser en digital o en analògica, ja que no tots els pins suporten les mateixes opcions d'entrada o sortida o recepció de senyals.
Per a cada funció, es va associar un botó en RF i un altre normal. Finalment, es va dissenyar el cos principal amb crides a aquestes funcions.

A la següent figura es pot veure un diagrama de flux del programa principal.

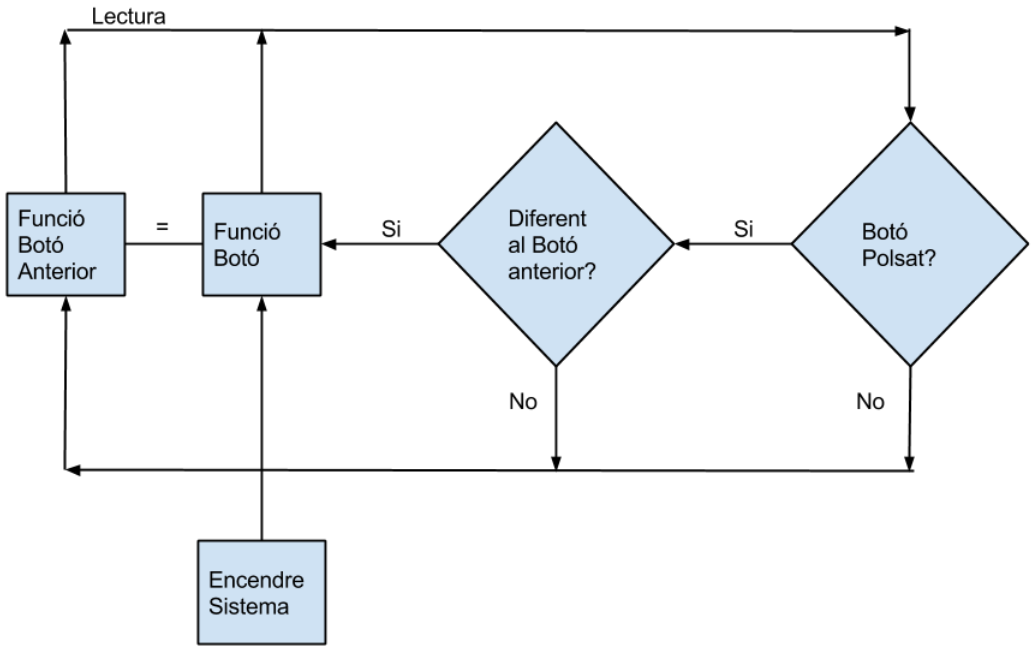


Figura 3: Diagrama de Flux del Funcionament del PIC

A la figura anterior es veu com, per començar, s'ha d'encendre el sistema. Només encendre's, el sistema arrenca amb una funció base on restarà esperant a que els diferents botons siguin polsats. En el cas que no es polsin, seguirà fent la seva funció indefinidament.
En el cas que algun botó sigui polsat, el sistema s'actualitzarà i passarà a fer la funció corresponent, excepte si el botó polsat correspon a la mateixa funció que ja està corrent. En aquest cas el botó no produirà cap efecte.

```
void main(void)
{
   init_ports();
   while(1)
   {
       if((PORTCbits.RC4) || (PORTCbits.RC1)) //Apretamos el boton1
       {
           //
           rele_Motor();
       }
       else if ((PORTDbits.RD3) || (PORTCbits.RC2)) //Apretamos el boton2
       {
           //
           leds_Rojos();
       }
```

Figura 4: Codi del Main

4.3. Anàlisi de les Diferents Funcions

En termes generals, totes les funcions funcionen d'una manera similar i ara es farà un breu anàlisi d'elles.
El factor comú entre elles és la seva metodologia per a saber si s'està realitzant la funció correcta o no.

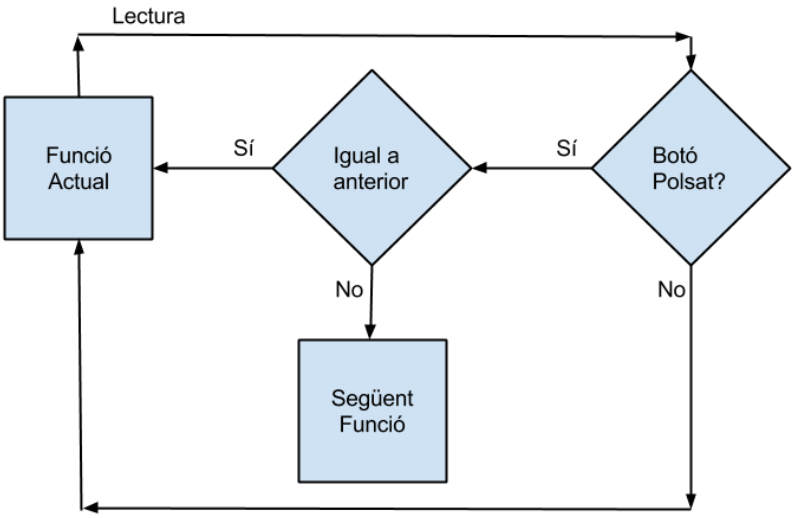


Figura 5: Diagrama de Flux d'una Funció

El diagrama anterior comença quan una funció escollida s'està realitzant. Mentre aquesta funciona, realitzarà periòdicament lectures globals per veure si algun botó ha estat polsat. Si el botó s'ha polsat, mirarà si s'ha polsat el botó propi de la funció que està corrent. Si el botó no es l'últim que s'ha polsat, aleshores es farà la funció corresponent.
Encendre el LED es fa de forma digital i des del software programat. Això vol dir que el LED o estarà encès o estarà apagat.
Hi ha funcions que en contes d'un sistema digital, es vol utilitzar una versió més analògica del LED fent un increment i un decrement de la intensitat de la llum emesa pel LED. Això s'ha aconseguit fent pampalluguejar el LED amb més o menys intensitat de manera que la mitja que l'ull humà veu sigui més o menys intensa, d'aquesta manera s'aconsegueix un efecte molt més fluid i no tant molest per la persona.
Aquest tipus de funció pot ser la del mode automàtic.

4.3.1. Activar el Relé del Motor de Bombolles

L'objectiu d'aquesta funció és activar el motor de bombolles del tub. Com el projecte es un prototip i no es disposa de cap motor de bombolles, va ser acceptada la opció que al prémer el botó 1, corresponent a aquesta funció, s'encengués un LED on en una versió més definitiva només caldrà connectar la sortida del LED al relé del motor.

```
void rele_Motor (void) //Activar el relé del motor de burbujas (boton1) - en este caso solamente e
{
    while(1)
    {
        PORTD = 0x20; //turn LED B

        if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) || (PORTAbits
        {
            break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
        }
    }
    return;
}
```

Figura 6: Codi per encendre el LED

4.3.2. Encendre LED Vermell

Com ja s'ha explicat anteriorment, el LED del projecte es un RGB, que està composat intrínsecament per 3 LEDs interns dels 3 colors vermell, verd i blau. Aquesta funció farà encendre el LED vermell traient la intensitat pel pin on està connectat el LED vermell i així s'encendrà.

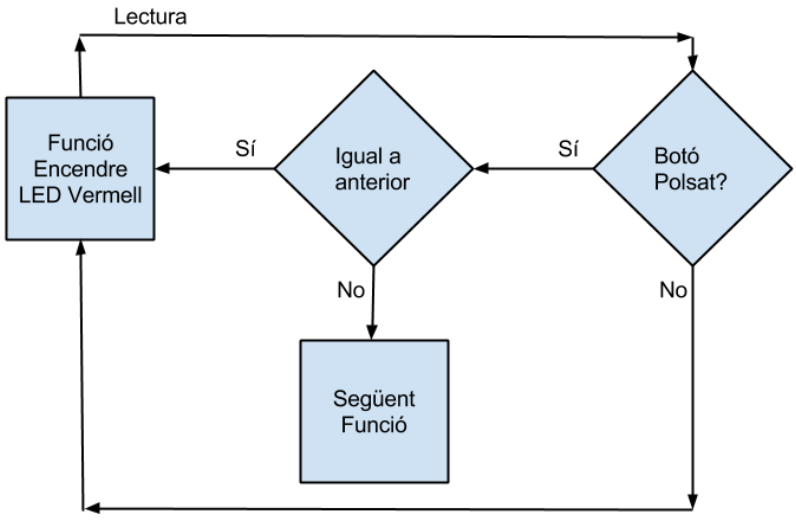


Figura 7: Funció LED Vermell

Com es pot veure a la Figura 3, un cop s'ha polsat el botó corresponent a la funció d'encendre el LED Vermell (botó 2) el PIC traurà una tensió pel pin corresponent que farà encendre el LED vermell. Un cop la funció ha encès el LED, farà una lectura per veure si algun botó, excloent el botó 2, està polsat. Si es així, sortirà de la funció 2 i anirà cap a la funció que correspongui. Si no s'ha polsat cap botó o s'ha polsat el botó 2, el LED seguirà encès.

Aquest procés es repeteix per les funcions d'encendre els LEDs Verd i Blau.
En el cas de les funcions per encendre el LED Groc i Magenta, el diagrama de flux seria el mateix però en contes de donar tensió per un pin es donaria tensió per dos pins simultàniament.

Així doncs, en el cas de voler encendre el LED Groc encendríem el LED Vermell i Verd; i si volguéssim encendre el LED Magenta, encendríem simultàniament el LED Vermell i Blau.

Per no haver de prémer dos botons a la vegada, s'ha creat una funció per cada botó que encendrà automàticament els 2 LEDs.

Pel cas de la funció d'apagar el LED, tornarà a tindre el mateix diagrama de flux de les funcions anteriors però en contes d'emetre voltatge per algun pin, no emetrà res.

4.3.3. Mode Automàtic

Aquest mode també està considerat funció. Correspon a la funció del botó 8 i, com ja s'ha explicat anteriorment, només encendre el Sistema es posa aquesta funció per defecte. Que sigui una funció d'un botó també permet que en el moment que es vulgui, es podrà reiniciar el programa.

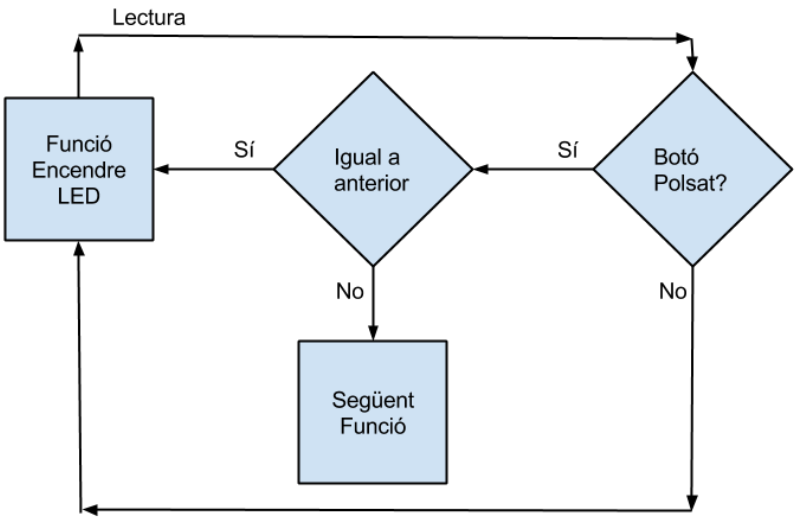


Figura 8: Flux del Mode Automàtic

En aquesta funció s'encendrà el LED durant aproximadament dos segons amb un batec constant encenent-se i apagant-se lentament, de major a menor intensitat i a la inversa, canviant de color cada vegada. Com a la resta de funcions, fins que no es polsi algun botó diferent al vuitè, el LED se seguirà encenent i canviant de color.

```
while(1)
{  //following loop increases the pwm duty cycle from 0% to 100% on portD
    for(signal_high=1; signal_high<100; signal_high++)
    {
        for(count=0; count<signal_high; count++)
        {
            PORTD = 0x80;
        }
        signal_low=100-signal_high;
        for(count=0; count<signal_low; count++)
        {
                PORTD = 0x00;
        }
    }
    if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) || (PORTAbits.RA
    {
        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
    }
```

Figura 9: Fragment de codi que fa bategar un LED

4.3.4. Mode Micròfon

El mode Micròfon correspon al botó 9, un cop polsat, el LED respondrà a la veu que escolti pel micròfon connectat. Cada cop que se senti un pic de veu, el LED canviarà de color.

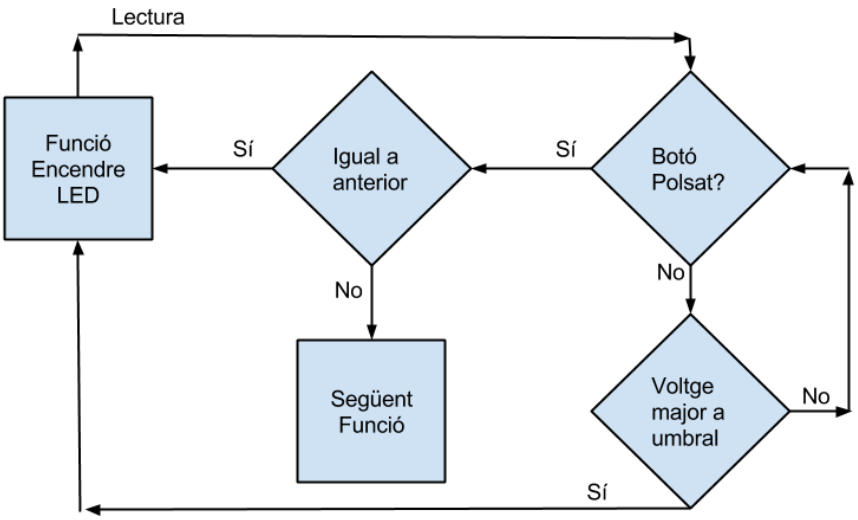


Figura 10: Flux del Mode Micròfon

La funció es molt similar a les funcions d'encendre els LEDs de colors, amb l'afegit que també es mirarà si el llindar està superat per la veu quan es parla pel micròfon.
El LED s'encendrà quan detecti que s'ha superat el llindar establert i es mantindrà encès sense apagar-se fins a canviar al següent color o canviar de funció.

La funció llegirà si s'ha polsat un botó diferent al 9 i, si es així, canviarà de funció, si no, mirarà si el voltatge resultant del micròfon supera cert Umbral i si es així, canviarà el color del LED, si no, tornarà a llegir si s'ha polsat el botó.

```
ADCON0 = 0x00;//clear ADCON0 to select channel 0 (AN0)
ADCON2 = 0b00001000;//ADCON2 setup: Left justified, Tacq=2Tad, Tad=2*Tosc (or Fosc/2)
ADCON0bits.ADON = 0x01;//Enable A/D module


     // configure A/D convertor
    OpenADC( ADC_FOSC_16      &
         ADC_RIGHT_JUST &
         ADC_12_TAD,
         ADC_CH0      &
         ADC_INT_OFF ,
         ADC_VREFPLUS_VDD);

    while (1)
    {
        ConvertADC(); // Start conversion
        while( BusyADC() ); // Wait for completion
        ADC_Output = ReadADC(); // Read result
        if (ADC_Output>128)
        {
            if(PORTDbits.RD7) //Si led rojo encendido, se enciende el verde
            {
                PORTD = 0x40;
            }
```

Figura 11: Part del codi del Micròfon

4.3.5. Mode SHX (DMX)

Aquest mode està composat pels 2 microcontroladors i ambdós tenen funcionalitats dintre del sistema que porta a la resposta definitiva d'il·luminació DMX.
El DMX al PIC, al no activar-se per botons, s'activarà quan detecti que rep senyal de DMX.
Aquesta senyal vindrà processada i dividida per l'Arduino en els 3 valors bàsics de la llum vermell, verd i blau (RGB) que entraran per 3 inputs diferents del PIC.

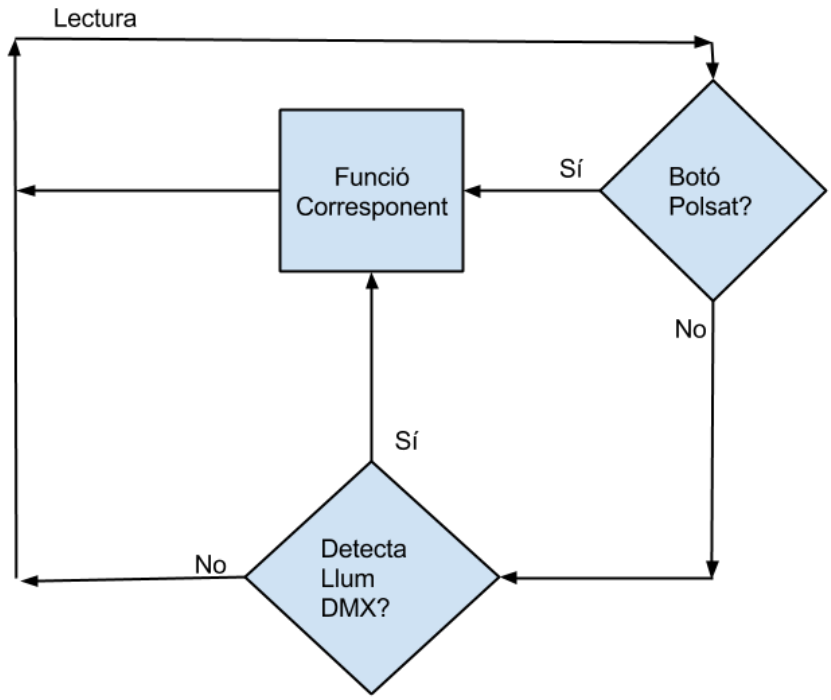


Figura 12: Diagrama de blocs DMX

Pel codi DMX hi ha 2 parts ja conegudes, el processat de la senyal per l'Arduino i la sortida en mode de llum degut al PIC. Es mostra un fragment corresponent a cada codi.

```
void leds_Dmx (void) //funcion procesado señal DMX
{
    //uint16_t adc_value;
    int ADC_OutputR, ADC_OutputG, ADC_OutputB;  // variable to hold ADC conversion result in

//pograma
    //TRISAbits.TRISA0 = 0;               // set pin as output
    TRISDbits.TRISD7 = 0;
    TRISAbits.TRISA1 = 1;               // set pin as input
    //ANCON0bits.ANSEL2= 0;               // set pin as digital

    while(1)
    {
        if((PORTAbits.RA1)&&(PORTAbits.RA2)&&(PORTAbits.RA3))  // if the pin is high
        {                                                       //led blanco
            LED_D7 = 1;
            LED_D6 = 1;
            LED_D5 = 1;
```

Figura 13: Codi DMX del PIC

4.4. Programar un Arduino

Un cop es té el software descarregat gratuïtament i configurat:

- Obrir (Editar)
- Verificar (Compilar)
- Pujar (Gravar)
- Provar el programa

El primer pas es obrir el codi prèviament guardat o editat amb l'objectiu que l'Arduino acabi fent el que se li ha demanat[11]. Aquest procés es el d'obrir o editar.

La programació s'ha fet amb el llenguatge del propi Arduino , amb l'ajuda de la llibreria DMX de 4 universos versió 0,3.

Per a compilar el PIC s'utilitza el botó verificar, mostrant els possibles errors i donant una resposta visible en el software si es demana.

Gravar o pujar el programa és el procés que passa d'una versió que entenen els humans a una versió més bàsica a llenguatge màquina on el que ho entén es el propi Arduino.

Finalment es testeja l'Arduino amb el hardware corresponent, en aquest cas els LEDs i s'espera que tingui la resposta desitjada. En el cas que no tingui una resposta correcta, tornarem al primer pas, a 'Editar' el codi.

4.4.1. Receptor de paquets DMX

L'Arduino UNO té unes entrades que es poden configurar com a de recepció de senyals i son perfectament vàlides per a les senyals DMX.
El que el codi realitza en aquest cas és descompondre la senyal que li arriba per un mateix univers seleccionat en tres canals diferents, on en cada canal hi ha el valor de la component del color fragmentada en vermell, verd i blau.
Divideix la senyal en 3 valors, un per canal, i treu els valors pels pins de sortida també seleccionats prèviament on el PIC els processarà.

```
void loop()
{
  for(;;)
  {
    //write values from dmx channels 1-4 universe 0 to arduino pwm pins 5-8
    analogWrite(5, ArduinoDmx0.RxBuffer[0]);  // led Rojo -- buffers 0 indexed
    analogWrite(6, ArduinoDmx0.RxBuffer[1]);  //led Verde
    analogWrite(7, ArduinoDmx0.RxBuffer[2]);  //led Azul
    //analogWrite(8, ArduinoDmx0.RxBuffer[3]);
  }
}  //end loop()
```

Figura 14: Codi DMX Arduino

Tal i com s'ha comentat anteriorment, el software està configurat per a que treballi amb un sol univers i llegeixi el senyal de 3 canals prèviament establerts. Aquesta configuració no es única, la diferència d'aquest codi respecta altres que s'ha trobat es que tant pot rebre senyal DMX des d'un univers com de 2 i barrejar les senyals, juntament amb els seus canals. També es pot establir prioritats per a un dels universos.
A la hora de transmetre senyal DMX, l'Arduino es pot programar per a que utilitzi interrupcions per a enviar o rebre de manera síncrona i pot utilitzar els modes DMX més ràpids que no estan estandarditzats.
Tot això es pot fer a partir de la llibreria implementada "lib_dmx.h" que té com a principals funcions per aquesta aplicació o possibles noves aplicacions o canvis:

- ArduinoDmxN.set_control_pin(PIN_NUMBER):

Cridant a aquesta funció s'assigna el número del pin que s'utilitzarà per a controlar el driver MAX1485, pot funcionar com univers d'entrada o de sortida.

- ArduinoDmxN.set_rx_address(DMX_RX_ADDRESS)

Aquesta funció assigna l'adreça d'entrada DMX per aquest univers. Pot ser qualsevol valor entre 1 i 512.

- ArduinoDmxN.set_tx_address(DMX_TX_ADDRESS);

Asigna l'adreça de sortida DMX per aquest univers podent ser qualsevol valor entre 1 i 512.

- ArduinoDmxN.set_rx_channels(NUMBER_OF_CHANNELS);

La funció assigna el nombre de canals d'entrada DMX per aquest univers. Pot ser qualsevol valor entre 2 i 512.

- ArduinoDmxN.set_tx_channels(NUMBER_OF_CHANNELS);

Assigna el nombre de canals de sortida DMX per aquest univers, podent agafar qualsevol valor entre 2 i 512.

- ArduinoDmxN.attachTXInterrupt(my_TX_ISR_name);

S'assigna el nom de la funció de callback que s'activa cada vegada que s'acaba d'enviar un tram DMX a l'univers. Evita pèrdua de temps al bucle principal del programa.

- ArduinoDmxN.attachRXInterrupt(my_RX_ISR_name);

Assigna el nom de la funció de callback que s'activa cada vegada que s'acaba de rebre un tram DMX a un univers.

- ArduinoDmxN.init_rx(DMX_MODE);

S'inicia l'univers per a rebre DMX un cop cridades les altres funcions.

Arduino DmxN.init_tx(DMX_MODE)

S'inicia l'univers per a transmetre DMX un cop cridades les altres funcions.

CAPÍTOL 5. VALIDACIÓ I PRESSUPOST

En aquest capítol s'explica quin procediment s'ha seguit per a comprovar el correcte funcionament de les diferents funcionalitats i, finalment, el cost que el prototip ha suposat exposant els diferents materials dels que està composat el sistema. Tal i com s'ha explicat a l'inici del projecte, es important que el sistema sigui assequible per a tothom.

Al prototip no s'ha utilitzat un LED RGB real sinó que s'ha utilitzat dos LEDs vermells i un verd perquè el funcionament en sí es equivalent.

Es connecta el circuit a una font d'alimentació de 5V, que alimentarà el PIC, el ADM1438, l'amplificador del micròfon i l'oscil·lador. També servirà per donar tensió als botons del tipus *Pull-Up Resistors*, on quan el botó sigui polsat, entrarà la tensió al pin del PIC corresponent.

5.1. Activar el relé del motor de bombolles

Com ja s'ha comentat anteriorment, al ser un prototip no es té el projecte connectat a un motor de bombolles real.
L'objectiu era que quan es polsés el botó corresponent, el motor s'activés. El que s'ha fet es encendre un LED com a equivalència real ja que la mateixa tensió de sortida podrà activar el relé i tindrà la mateixa funció.

Per a fer-ho possible es va buscar un codi equivalent per internet i es va adaptar al microcontrolador utilitzat, fent que la sortida fos pel pin correcte.
Es va utilitzar un oscil·loscopi per veure si rebia tensió la entrada del PIC i si s'obtenia la resposta correcta. Una senya continua en el temps.

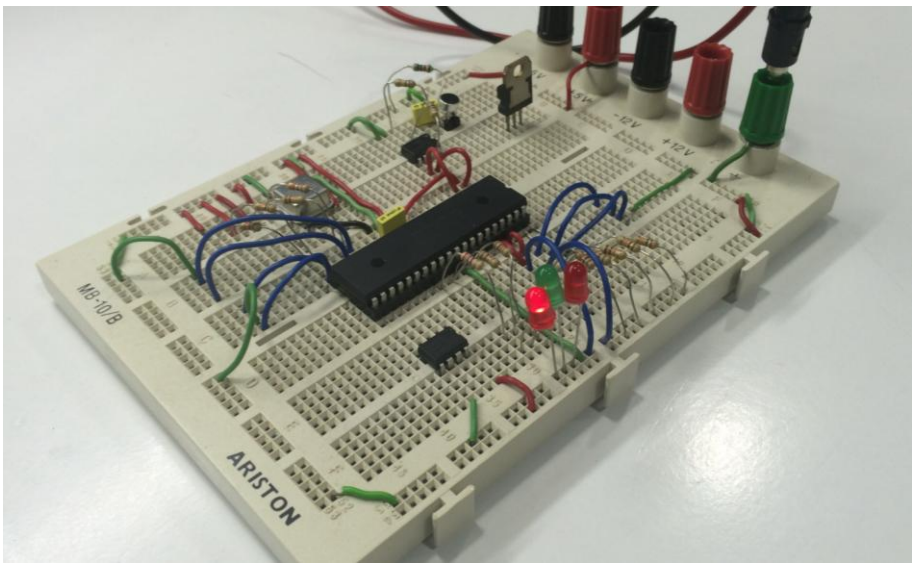


Figura 15: Il·luminació del LED corresponent a la funció

5.2. Encendre els LEDs

Pels casos d'encendre els diferents LEDs és similar al primer cas, en contes de rebre la tensió del botó per un port d'entrada, es polsarà el botó corresponent i entrarà pel port d'entrada corresponent programat i obtindrà la senyal pel port de sortida.

Per comprovar-ho es van polsar els diferents botons realitzant diferents combinacions de sortides. Tres sortides individuals i dues amb dues senyals a l'hora, fent encendre dos LEDs al mateix temps.

5.3. Encendre el LED Groc i Magenta

Per a encendre el LED groc i el magenta, com s'ha comentat anteriorment, necessitarem que s'encengui una combinació de dos LEDs per a aconseguir el color desitjat. En aquest cas els LEDs vermell i verd, per mostrar el groc o el vermell i el blau per a fer el magenta.

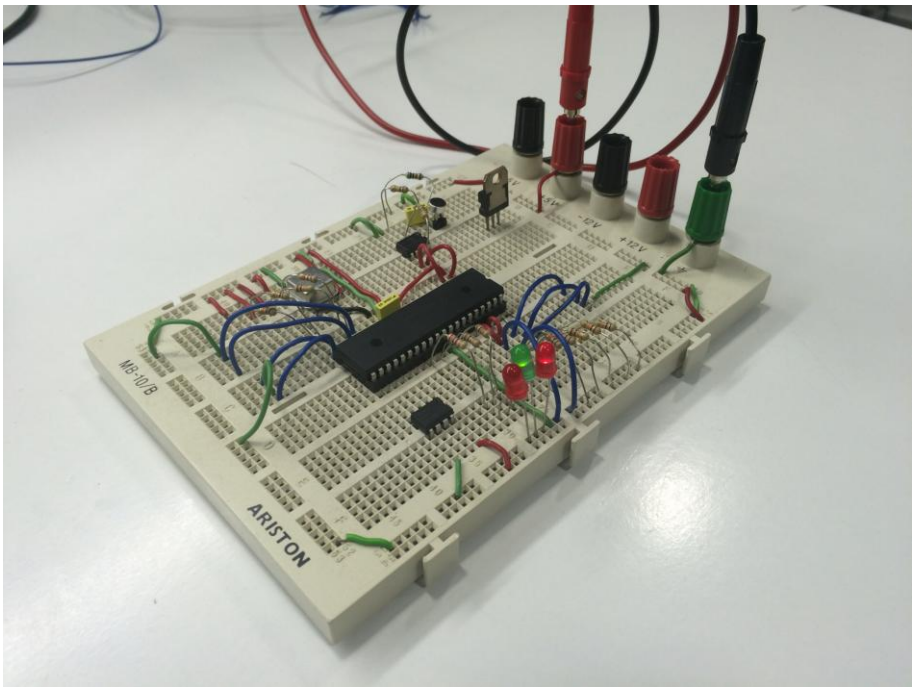


Figura 16: Il·luminació del LED corresponent a la funció

5.4. Apagar el llum

En aquest cas, les tres sortides es configuren com a apagades i no sortirà cap tensió per cap dels tres ports. En qualsevol situació de qualsevol funció, els LEDs, tan si estan encesos com si no, s'apagaran.

5.5. Mode Automàtic

El botó del mode automàtic genera una funció que es va repetint amb el temps. Els LEDs van bategant encenent-se i apagant-se lentament i canviant de color, passant per la gama de colors que es treballa.

Té una funcionalitat similar que amb el simple fet d'encendre els LEDs però en aquest cas es combina amb la funció que fa pampalluguejar el LED a major o menor freqüència simulant que està apagat o encès amb una intensitat major o menor respectivament.

5.6. Mode Micròfon

El mode micròfon també està associat a un botó, en aquest cas el novè, a diferència dels altres, el botó desbloqueja la opció de rebre per la entrada del micròfon i apropant-se al micròfon i parlant en veu alta, s'aconsegueix que passat un llindar de to elevat, els LEDs vagin canviant de color.

Per a comprovar el correcte funcionament es va dissenyar un amplificador. Tant a la entrada com a la sortida es van posar les sondes dels 2 canals de l'oscil·loscopi per veure la evolució del senyal. Vam ajustar el senyal a 2,5V com a llindar per veure clarament la senyal que provocava la veu.

5.7. Mode SHX (DMX)

Aquest mode no funcionarà en sí amb un botó, sinó amb un selector.
El senyal es crea al PC amb el Software que BJ Live proporciona, creant el color desitjat.

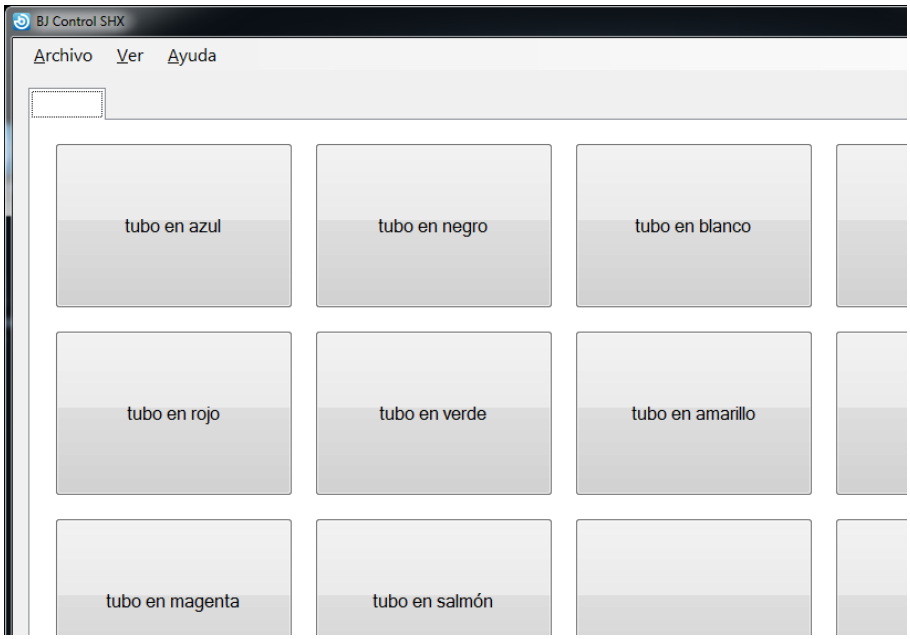


Figura 17: Software DMX de BJ.

Per USB i utilitzant el dispositiu DMX, anteriorment mencionat, transforma el senyal amb llum. Aquesta senyal passa pel ADM1485 per a que la senyal estigui entre uns valors determinats. Aquesta senyal entra al Arduino on hi ha el programa que treu els valors dels 3 canals seleccionats pels 3 ports diferents. El PIC detecta aquestes 3 senyals i encén els LEDs corresponents.

Per a la comprovació de la recepció es va col·locar la sonda abans i després del ADM i abans i després de l'Arduino per comprovar que realment donaven valor desitjat, entrant els valors primaris de blau, verd i vermell per separat i comprovant que realment només sortia la senyal corresponent per un dels 3 ports.
També es va arribar a comprovar amb una barreja de 2 colors com el color magenta, cap color, com el negre o tots els colors per igual, que resultaria ser el color blanc. En la barreja de colors amb diferent intensitat com per exemple el color salmó, donava una sortida de senyal més baixa pel port corresponent, essent igual a la resta de valors pels casos anteriorment parlats.

5.8. Mode Radiofreqüència

Com a objectiu també es volia poder controlar tot amb un comandament a distància, així que BJ va proporcionar el seu software i el seu hardware per a realitzar més eficientment la connexió entre el seu comandament a distància i la placa base.
El comandament a distancia funciona amb bateries i a una freqüència de 433 MHz, el receptor està situat a la placa base també creada per BJ, on interconnectant el cablejat amb la placa s'aconsegueix detectar quin botó s'ha polsat.

Quan es polsa un botó, un port de la placa receptora dona un voltatge de 5V, el qual aprofitem per posar com a senyal a un port d'entrada del PIC i configurar aquell port amb la funció desitjada.
Es va col·locar l'oscil·lador a cada un dels ports de sortida de la placa receptora per saber quin pin corresponia a cada botó del comandament i a partir d'allà es va configurar la resta.

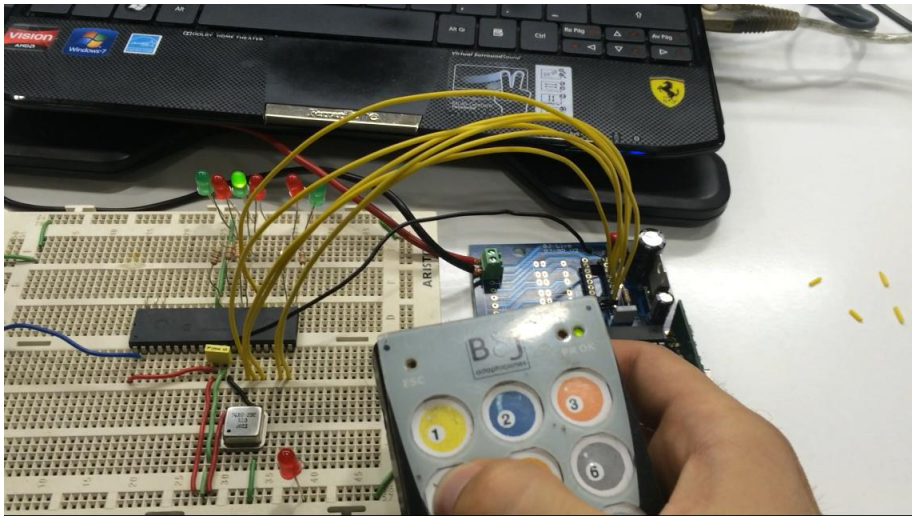


Figura 18: Al polsar el botó el LED s'encén.

5.9. Pressupost

Com s'ha indicat a l'inici del projecte, un dels objectius principals era aconseguir el mínim cost possible per aquest producte. Com s'ha vist en capítols anteriors, el cost elevat dels elements existents limita l'accés als usuaris amb discapacitats a aquest tipus de productes o a les empreses a poder-ne aconseguir un. A la següent taula es mostra el material utilitzat per a realitzar el projecte. [12]

Material	Preu unitari (€)	Quantitat	Preu total (€)
LED RGB	5,84	1	5,84
PIC18F4550	5,85	1	5,85
Oscil·lador 16 MHz	7,67	1	7,67
Amplificador LM386N-3	1,03	1	1,03
Resistències 10 KΩ	0,638	10	6,38
Resistències 270 Ω	1,26	3	3,78
Resistència 1 MΩ	3,82	1	3,82
Resistència 100KΩ	1,14	1	1,14
Condensador 100 nF	0,622	1	0,622
Condensador 150 nF	0,274	1	0,274
ADM1485 (RS485)	3,60	1	3,60
Micròfon	0,994	1	0,994
Arduino UNO	20	1	20

TOTAL	**54,097 €**

Tabla 8: Pressupost desglosat.

Com es pot veure en el pressupost un dels materials que més encareix el producte es la placa Arduino. Tot i així, si es realitza tal i com està previst, una matriu de LEDs RGB depenent de lo gran que sigui la matriu també encarirà el preu si es vol realitzar a mà però s'abaratirà molt si es compra a un proveïdor especialitzat.
Per exemple si volem realitzar una matriu de 16x32 LEDs, hi ha un total de 512 LEDs que a 5 € cada un suposen uns 2500 €, en canvi s'ha trobat matrius al voltant dels 40 € [13].

El fet que el producte sortís més barat si es fabriqués a gran escala no es un factor que el puguem tindre en compte ja que com hem especificat a l'inici, els usuaris son tan diversos que necessiten equips pràcticament personalitzats, tot i així aquest producte s'ha fet el màxim polivalent possible per a realitzar les màximes produccions.

CAPÍTOL 6. CONCLUSIONS I FUTURES AMPLIACIONS

A l'inici d'aquest projecte es van establir uns objectius que s'havien de complir per donar el projecte per satisfactori. A més, es buscarien els components i les característiques més òptimes per a que el cost fos el més baix possible.

Fins al dia d'avui hi havia el tub de bombolles Spacekraft i el tub de bombolles de BJ. El que es pretenia amb aquest projecte era unir aquestes dues idees de tubs de bombolles amb un millor cost amb la millora que es pogués seleccionar entre un funcionament autònom i el funcionament via SHX (DMX).
Aquest últim funcionament ja ve dissenyat i configurat i només afecta al projecte a l'hora de rebre el senyal, que es processarà i donarà senyal per a que s'encenguin els LEDs corresponents.
Dins del mode autònom, s'havien de complir 9 funcionalitats, corresponent als 9 botons, amb el requisit que només encendre el tub, s'encengués amb el mode automàtic.

Al prémer el primer botó, tant per botó cablejat com pel no cablejat, donarà senyal per encendre el relé del motor de bombolles.

El segon botó, dóna voltatge per encendre els LEDs de color vermell, el tercer de color verd, blau pel quart botó, groc pel cinquè, magenta pel sisè botó i el setè botó apagaria els LEDs.
Tots els botons es poden prémer tantes vegades com es vulgui i faran la funcionalitat que toca. Si un mateix botó es prem moltes vegades, el botó no es reiniciarà sinó que seguirà realitzant la tasca. Això està pensat per aquells que no podrien prémer el botó nomes un cop o si s'equivoquen de botó no succeeixi res especial.
Els dos últims botons son en sí modes de funcionament.
El vuitè botó es el mode automàtic, que està posat tant al arrancar el tub com si un el vol prémer. El novè i últim botó, es el mode micròfon que passa a llegir una entrada analògica d'àudio de veu on, en un principi, ha de canviar de color a cada crit de veu que senti.

Per a que el circuit esdevingui més estable, es podria soldar i passar a placa fixa per a que quedi a una mida més reduïda i es pugui arribar a vendre.
De la manera que s'ha dissenyat, el volum de LEDs es ampliable a la oferta o necessitats que es demanin.
Per tal de dur a terme això, es necessitaria un adaptador de potència que faria que la il·luminació dels LEDs no disminuís a mesura que es van introduint més quantitat de LEDs.
Aquest dispositiu també facilitaria l'especificació de la il·luminació dels LEDs en valors més intel·ligibles.
Recordem que el PIC es reprogramable i, al haver sobrat pins d'entrada i de sortida, es podrien col·locar i programar noves funcionalitats encara en el

mateix microcontrolador, de manera que s'estalviaria en material tant nou com de recanvi.
Qualsevol problema que passés amb el micròfon, al no tindre res a veure amb el projecte es podria connectar un altre ja que el que fa falta només es un que compleixi les condicions de tindre una sortida de jack d'àudio.
Es podria fer una instal·lació del projecte a una sala multi sensorial on els infants poguessin aprendre i comunicar-se amb l'adult i l'adult amb l'alumne.
A part d'estimular l'alumne amb la visió, l'audició i el tacte, també es podria ampliar al gust i a l'olfacte, a part de treballar també la relaxació, la elecció i la comunicació.
Aquestes sales multi sensorials es podrien també adaptar, no només per als infants sinó també per a la gent gran amb malalties mentals com, per exemple, l'Alzheimer que podrien servir per ajudar a exercitar la ment, relacionar-se i tindre estímuls per alentir la malaltia o prevenir-ne l'aparició.

REFERÈNCIES

[1] http://diversidad.murciaeduca.es/publicaciones/cee/doc/5_7.pdf (última revisió 19/08/2015)
[2] http://www.fem.es/Imatges/Web/Documents/Dificultades%20cognitivas.PDF (última revisió 19/08/2015)
[3] http://www.asperger.es/publicaciones_detalle.php?id=36&titulo=S%EDndrome%20de%20Asperger:%20Aspectos%20discapacitantes%20y%20Valoraci%F3n (última revisió 19/08/2015)
[4] http://www.neurologia.com/pdf/Web/5910/bm100443.pdf (última revisió 19/08/2015)
[5] http://www.bj-adaptaciones.com/ (última revisió 19/08/2015)
[6] http://www.geriatricasc.es/tienda/terapia-ocupacional-c-6.html (última revisió 19/08/2015)
[7] http://ca.wikipedia.org/wiki/D%C3%ADode_emissor_de_llum (última revisió 19/08/2015)
[8] http://ca.wikipedia.org/wiki/Microcontrolador (última revisió 19/08/2015)
[9] http://www.deskontrol.net (última revisió 19/08/2015)
[10] http://es.wikipedia.org/wiki/Digital_Multiplex (última revisió 19/08/2015)
[11] http://www.arduino.cc (última revisió 19/08/2015)
[12] http://es.farnell.com/ (última revisió 22/08/2015)
[13] http://tienda.bricogeek.com/

Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

ANNEXOS

Tub de Bombolles Autònom/SHX

TITULACIÓ: Grau en Enginyeria de Sistemes de Telecomunicació

AUTOR: Albert Martret Torrent

DIRECTOR: Jaime Oscar Casas Piedrafita

DATA: 2 de setembre de 2015

ANNEX I. CODIFICACIÓ PIC18F4550

```
/*
 * File:   main.c
 * Author: Albert Martret
 *
 * Consideraciones especiales: Proyecto Fina de Carrera, se usa con circuito protoboard externo.
 * Creado en Septiembre de 2015, 10:06
 */

/*
#####################################################################
**** PIC18F4550 LED Blinky
**** IDE : MPLAB Ver 8.8
**** Compiler : Microchip XC8
#####################################################################
*/

// Types
 typedef unsigned short U8;
 typedef unsigned int   U16;
// Includes
 #include <p18f4550.h>
 #include <delays.h>
 #include <timers.h>
 #include "pwm.h"
 #include <delays.h>
 #include <adc.h>
 #include <xc.h>
 #include <pwm.h>
 #include <stdio.h>

 #define LFSR(n)    {if (n&1) n=((n^0x80000055)>>1)|0x80000000; else n>>=1;}
 #define ROT(x, y)  (x=(x<<y)|(x>>(32-y)))

 // Configurations
 #pragma config WDT = OFF // Watchdog Timer Disabled
// #pragma config OSC = HS // OSC Mode - High Speed Crystal/Resonator
 #pragma config XINST = OFF // Extended Instructions Disabled

 // LEDs un total de 33. Doy nombres de LEDs a los pins:
 #define LED_E3 LATAbits.LATA3 //PIN 1 només entrada (usado como ref 5V)
 #define LED_A0 LATAbits.LATA0 //PIN 2 la resta es entrada/sortida
 #define LED_A1 LATAbits.LATA1 //PIN 3
 #define LED_A2 LATAbits.LATA2 //PIN 4
 #define LED_A3 LATAbits.LATA3 //PIN 5
 #define LED_A4 LATAbits.LATA4 //PIN 6
 #define LED_A5 LATAbits.LATA5 //PIN 7
 #define LED_E0 LATEbits.LATE0 //PIN 8
 #define LED_E1 LATEbits.LATE1 //PIN 9
 #define LED_E2 LATEbits.LATE2 //PIN 10
 #define LED_C0 LATCbits.LATC0 //PIN 15
 #define LED_C1 LATCbits.LATC1 //PIN 16
 #define LED_C2 LATCbits.LATC2 //PIN 17
 #define LED_D0 LATDbits.LATD0 //PIN 19
 #define LED_D1 LATDbits.LATD1 //PIN 20
 #define LED_D2 LATDbits.LATD2 //PIN 21
 #define LED_D3 LATDbits.LATD3 //PIN 22
 #define LED_C4 LATCbits.LATC4 //PIN 23
 #define LED_C5 LATCbits.LATC5 //PIN 24
 #define LED_C6 LATCbits.LATC6 //PIN 25
 #define LED_C7 LATCbits.LATC7 //PIN 26
 #define LED_D4 LATDbits.LATD4 //PIN 27
 #define LED_D5 LATDbits.LATD5 //PIN 28
 #define LED_D6 LATDbits.LATD6 //PIN 29
 #define LED_D7 LATDbits.LATD7 //PIN 30
 #define LED_B0 LATBbits.LATB0 //PIN 33
 #define LED_B1 LATBbits.LATB1 //PIN 34
 #define LED_B2 LATBbits.LATB2 //PIN 35
 #define LED_B3 LATBbits.LATB3 //PIN 36
 #define LED_B4 LATBbits.LATB4 //PIN 37
 #define LED_B5 LATBbits.LATB5 //PIN 38
 #define LED_B6 LATBbits.LATB6 //PIN 39
 #define LED_B7 LATBbits.LATB7 //PIN 40
```

```
////////////////////////////////////////////////////////////////////////////
void rele_Motor (void) //Activar el relé del motor de burbujas (boton1) - en este caso solamente encenderá un LED ya que no está conectado el relé
{
	while(1)
	{
		PORTD = 0x20; //turn LED B

		if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) || (PORTAbits.RA5) ||
			(PORTEbits.RE0) || (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC2) ||
			(PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) || (PORTBbits.RB3) ||
			(PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
		{
			break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
		}
	}
	return;
}

////////////////////////////////////////////////////////////////////////////
void leds_Rojos (void) //Enciende LED Rojo (boton2)
{
	while(1)
	{
		PORTD = 0x80; // turn LED R

		if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) || (PORTAbits.RA5) ||
			(PORTEbits.RE0) || (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
			(PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTCbits.RC4) || (PORTBbits.RB3) ||
			(PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
		{
			break; //pulsando cualquier botón, sale del bucle
		}
	}
	return;
}

////////////////////////////////////////////////////////////////////////////
void leds_Verdes (void) //Enciende LED Verde (boton3)
{
	while(1)
	{
		PORTD = 0x40; // turn LED G

		if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) || (PORTAbits.RA5) ||
			(PORTEbits.RE0) || (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
			(PORTCbits.RC2) || (PORTDbits.RD1) || (PORTDbits.RD3) || (PORTCbits.RC4) || (PORTBbits.RB3) ||
			(PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
		{
			break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
		}
	}
	return;
}

////////////////////////////////////////////////////////////////////////////
void leds_Azules (void) //Enciende LED Azul (boton4)
{
	while(1)
	{
		PORTD = 0x20; // turn LED B

		if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA5) || (PORTEbits.RE0) ||
			(PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) || (PORTCbits.RC2) ||
```

```
                (PORTDbits.RD0) || (PORTDbits.RD2) || (PORTDbits.RD3) || (PORTCbits.RC4) ||
(PORTBbits.RB3) ||
                (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
        {
                break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
        }
    }
    return;
}

////////////////////////////////////////////////////////////////////////////
void leds_Amarillos (void) //Enciende LED Amarillo (boton5)
{
    while(1)
    {
        PORTD = 0xC0; // turn LED R & G

        if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTEbits.RE0) ||
                (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
        {
                break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
        }
    }
    return;
}

////////////////////////////////////////////////////////////////////////////
void leds_Magenta (void) //Enciende LED Magenta (boton6)
{
    while(1)
    {
        PORTD = 0xA0; // turn LED R & B

        if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                (PORTBbits.RB3) || (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
        {
                break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
        }
    }
    return;
}

////////////////////////////////////////////////////////////////////////////
void apaga_Leds (void) //Apaga el LED (boton7)
{
    while(1)
    {
        PORTD = 0x00; // turn OFF LEDs

        if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                (PORTEbits.RE0) || (PORTEbits.RE2) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB6) || (PORTBbits.RB7))
        {
                break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
        }
    }
    return;
}

////////////////////////////////////////////////////////////////////////////
void Microfono (void) //Modo Micrófono - cambia el color cada vez que la señal supera un nivel (boton 9)
{
```

```
	int ADC_Output=0;

//	#pragma config FOSC = HS
	#pragma config LVP = OFF /* Turns Low Voltage Programming off */
	#pragma config MCLRE = ON /* To enable MCLR - To enable PIC reset */

	#pragma config FOSC = INTOSCIO_EC //Internal oscillator, port function on RA6, EC used by USB
	#pragma config WDT = OFF //Disable watchdog timer

	ADCON0 = 0x00;//clear ADCON0 to select channel 0 (AN0)
	ADCON2 = 0b00001000;//ADCON2 setup: Left justified, Tacq=2Tad, Tad=2*Tosc (or Fosc/2)
	ADCON0bits.ADON = 0x01;//Enable A/D module


		// configure A/D convertor
	  OpenADC( ADC_FOSC_16     &
	     ADC_RIGHT_JUST &
	     ADC_12_TAD,
	     ADC_CH0     &
	     ADC_INT_OFF ,
	     ADC_VREFPLUS_VDD);

	  while (1)
	  {
	     ConvertADC(); // Start conversion
	     while( BusyADC() ); // Wait for completion
	     ADC_Output = ReadADC(); // Read result
	     if (ADC_Output>128)
				{
		    if(PORTDbits.RD7) //Si led rojo encendido, se enciende el verde
					{
						PORTD = 0x40;
					}
					if(PORTDbits.RD6) //Si led verde encendido, se enciende el azul
					{
						PORTD = 0x20;
					}
					if(PORTDbits.RD5) //Si led azul encendido, se enciende el amarillo
					{
						PORTD = 0xC0;
					}
					if((PORTDbits.RD7)&&(PORTDbits.RD6)) //Si led amarillo encendido, se
enciende el cyan
					{
						PORTD = 0x60;
					}
					if((PORTDbits.RD6)&&(PORTDbits.RD5)) //Si led cyan encendido, se enciende
el magenta
					{
						PORTD = 0xA0;
					}
					if ((PORTDbits.RD7)&&(PORTDbits.RD5)) //Si led magenta encendido, se
enciende el rojo
					{
						PORTD = 0x80;
					}
		}

				if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
					(PORTEbits.RE0) || (PORTEbits.RE1) || (PORTEbits.RE2) || (PORTCbits.RC1)
|| (PORTCbits.RC2) ||
					(PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3)
|| (PORTCbits.RC4) ||
					(PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5)  || (PORTBbits.RB6))
				{
					break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
				}
	    }
	return;
}

////////////////////////////////////////////////////////////////////////////////
void Arrancar(void) //Al arrancar se pone en Modo Automático (boton8) los leds parpadean
{
	//Delay10TCYx (60); // gives a delay of 10 x 60 x 1/12 = 600/12 = 50 us
```

```
        long int count;
        long int i;
        int signal_high;
        int signal_low;
        TRISD = 0x00;

        while(1)
        {  //following loop increases the pwm duty cycle from 0% to 100% on portD
                for(signal_high=1; signal_high<100; signal_high++)
    {
       for(count=0; count<signal_high; count++)
       {
          PORTD = 0x80;
       }
       signal_low=100-signal_high;
       for(count=0; count<signal_low; count++)
       {
            PORTD = 0x00;
       }
                }
                if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                        (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                }
                //following loop decreases the pwm duty cycle from 100% to 0% on portD
                for(signal_high=1; signal_high<100; signal_high++)
                {
                        for(count=0; count<signal_high; count++)
                        {
                                PORTD = 0x00;
                        }
                        signal_low=100-signal_high;
                        for(count=0; count<signal_low; count++)
                        {
                                PORTD = 0x80;
                        }
                }
                if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                        (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                }

                for(signal_high=1; signal_high<100; signal_high++)
    {
       for(count=0; count<signal_high; count++)
       {
          PORTD = 0x40;
       }
       signal_low=100-signal_high;
       for(count=0; count<signal_low; count++)
       {
            PORTD = 0x00;
       }
                }
                if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                        (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
```

```
                    }
                    //following loop decreases the pwm duty cycle from 100% to 0% on portD
                    for(signal_high=1; signal_high<100; signal_high++)
                    {
                            for(count=0; count<signal_high; count++)
                            {
                                    PORTD = 0x00;
                            }
                            signal_low=100-signal_high;
                            for(count=0; count<signal_low; count++)
                            {
                                    PORTD = 0x40;
                            }
                    }
                    if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                            (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                            (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                            (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                    {
                            break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    for(signal_high=1; signal_high<100; signal_high++)
    {
       for(count=0; count<signal_high; count++)
       {
          PORTD = 0x20;
       }
       signal_low=100-signal_high;
       for(count=0; count<signal_low; count++)
       {
            PORTD = 0x00;
       }
                    }
                    if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                            (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                            (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                            (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                    {
                            break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    //following loop decreases the pwm duty cycle from 100% to 0% on portD
                    for(signal_high=1; signal_high<100; signal_high++)
                    {
                            for(count=0; count<signal_high; count++)
                            {
                                    PORTD = 0x00;
                            }
                            signal_low=100-signal_high;
                            for(count=0; count<signal_low; count++)
                            {
                                    PORTD = 0x20;
                            }
                    }
                    if ((PORTAbits.RA1) || (PORTAbits.RA2) || (PORTAbits.RA3) || (PORTAbits.RA4) ||
(PORTAbits.RA5) ||
                            (PORTEbits.RE0) || (PORTEbits.RE1) || (PORTCbits.RC0) || (PORTCbits.RC1) ||
(PORTCbits.RC2) ||
                            (PORTDbits.RD0) || (PORTDbits.RD1) || (PORTDbits.RD2) || (PORTDbits.RD3) ||
(PORTCbits.RC4) ||
                            (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                    {
                            break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    for(signal_high=1; signal_high<100; signal_high++)
    {
       for(count=0; count<signal_high; count++)
       {
          PORTD = 0xC0;
       }
       signal_low=100-signal_high;
```

```
        for(count=0; count<signal_low; count++)
        {
              PORTD = 0x00;
        }
                    }
                    if   ((PORTAbits.RA1)   ||   (PORTAbits.RA2)   ||   (PORTAbits.RA3)   ||   (PORTAbits.RA4)   ||
(PORTAbits.RA5) ||
                              (PORTEbits.RE0)   ||   (PORTEbits.RE1)   ||   (PORTCbits.RC0)   ||   (PORTCbits.RC1)   ||
(PORTCbits.RC2) ||
                              (PORTDbits.RD0)   ||   (PORTDbits.RD1)   ||   (PORTDbits.RD2)   ||   (PORTDbits.RD3)   ||
(PORTCbits.RC4) ||
                              (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                    {
                              break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    //following loop decreases the pwm duty cycle from 100% to 0% on portD
                    for(signal_high=1; signal_high<100; signal_high++)
                    {
                              for(count=0; count<signal_high; count++)
                              {
                                        PORTD = 0x00;
                              }
                              signal_low=100-signal_high;
                              for(count=0; count<signal_low; count++)
                              {
                                        PORTD = 0xC0;
                              }
                    }
                    if   ((PORTAbits.RA1)   ||   (PORTAbits.RA2)   ||   (PORTAbits.RA3)   ||   (PORTAbits.RA4)   ||
(PORTAbits.RA5) ||
                              (PORTEbits.RE0)   ||   (PORTEbits.RE1)   ||   (PORTCbits.RC0)   ||   (PORTCbits.RC1)   ||
(PORTCbits.RC2) ||
                              (PORTDbits.RD0)   ||   (PORTDbits.RD1)   ||   (PORTDbits.RD2)   ||   (PORTDbits.RD3)   ||
(PORTCbits.RC4) ||
                              (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                    {
                              break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    for(signal_high=1; signal_high<100; signal_high++)
     {
        for(count=0; count<signal_high; count++)
        {
           PORTD = 0x60;
        }
        signal_low=100-signal_high;
        for(count=0; count<signal_low; count++)
        {
              PORTD = 0x00;
        }
                    }
                    if   ((PORTAbits.RA1)   ||   (PORTAbits.RA2)   ||   (PORTAbits.RA3)   ||   (PORTAbits.RA4)   ||
(PORTAbits.RA5) ||
                              (PORTEbits.RE0)   ||   (PORTEbits.RE1)   ||   (PORTCbits.RC0)   ||   (PORTCbits.RC1)   ||
(PORTCbits.RC2) ||
                              (PORTDbits.RD0)   ||   (PORTDbits.RD1)   ||   (PORTDbits.RD2)   ||   (PORTDbits.RD3)   ||
(PORTCbits.RC4) ||
                              (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB6) || (PORTBbits.RB7))
                    {
                              break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
                    //following loop decreases the pwm duty cycle from 100% to 0% on portD
                    for(signal_high=1; signal_high<100; signal_high++)
                    {
                              for(count=0; count<signal_high; count++)
                              {
                                        PORTD = 0x00;
                              }
                              signal_low=100-signal_high;
                              for(count=0; count<signal_low; count++)
                              {
                                        PORTD = 0x60;
                              }
                    }
                    if   ((PORTAbits.RA1)   ||   (PORTAbits.RA2)   ||   (PORTAbits.RA3)   ||   (PORTAbits.RA4)   ||
(PORTAbits.RA5) ||
```

```
                        (PORTEbits.RE0)  ||  (PORTEbits.RE1)  ||  (PORTCbits.RC0)  ||  (PORTCbits.RC1)  ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0)  ||  (PORTDbits.RD1)  ||  (PORTDbits.RD2)  ||  (PORTDbits.RD3)  ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                }
                for(signal_high=1; signal_high<100; signal_high++)
    {
       for(count=0; count<signal_high; count++)
       {
          PORTD = 0xA0;
       }
       signal_low=100-signal_high;
       for(count=0; count<signal_low; count++)
       {
            PORTD = 0x00;
       }
                }
                if  ((PORTAbits.RA1)  ||  (PORTAbits.RA2)  ||  (PORTAbits.RA3)  ||  (PORTAbits.RA4)  ||
(PORTAbits.RA5) ||
                        (PORTEbits.RE0)  ||  (PORTEbits.RE1)  ||  (PORTCbits.RC0)  ||  (PORTCbits.RC1)  ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0)  ||  (PORTDbits.RD1)  ||  (PORTDbits.RD2)  ||  (PORTDbits.RD3)  ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                }
                //following loop decreases the pwm duty cycle from 100% to 0% on portD
                for(signal_high=1; signal_high<100; signal_high++)
                {
                        for(count=0; count<signal_high; count++)
                        {
                                PORTD = 0x00;
                        }
                        signal_low=100-signal_high;
                        for(count=0; count<signal_low; count++)
                        {
                                PORTD = 0xA0;
                        }
                }
                if  ((PORTAbits.RA1)  ||  (PORTAbits.RA2)  ||  (PORTAbits.RA3)  ||  (PORTAbits.RA4)  ||
(PORTAbits.RA5) ||
                        (PORTEbits.RE0)  ||  (PORTEbits.RE1)  ||  (PORTCbits.RC0)  ||  (PORTCbits.RC1)  ||
(PORTCbits.RC2) ||
                        (PORTDbits.RD0)  ||  (PORTDbits.RD1)  ||  (PORTDbits.RD2)  ||  (PORTDbits.RD3)  ||
(PORTCbits.RC4) ||
                        (PORTBbits.RB3) || (PORTBbits.RB4) || (PORTBbits.RB5) || (PORTBbits.RB7))
                {
                        break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                }
        }
        return;
}

////////////////////////////////////////////////////////////////////////////////
void leds_Dmx (void) //funcion procesado señal DMX
{
        //uint16_t adc_value;
        int ADC_OutputR, ADC_OutputG, ADC_OutputB; // variable to hold ADC conversion result in

//programa
        //TRISAbits.TRISA0 = 0;           // set pin as output
        TRISDbits.TRISD7 = 0;
   TRISAbits.TRISA1 = 1;           // set pin as input
   //ANCON0bits.ANSEL2= 0;           // set pin as digital

        while(1)
   {
      if((PORTAbits.RA1)&&(PORTAbits.RA2)&&(PORTAbits.RA3))  // if the pin is high
      {
                //led blanco
                        LED_D7 = 1;
                        LED_D6 = 1;
```

```
                                LED_D5 = 1;
        //LATAbits.LATA0 = 1;       // set the pin as high
    }
                    if((PORTAbits.RA1)&&(PORTAbits.RA2))            // if the pin is high
    {
        //led amarillo
                                LED_D7 = 1;
                                LED_D6 = 1;
                                LED_D5 = 0;
    }
                    if((PORTAbits.RA1)&&(PORTAbits.RA3))            // if the pin is high
    {
        //led magenta
                                LED_D7 = 1;
                                LED_D6 = 0;
                                LED_D5 = 1;
        //LATAbits.LATA0 = 1;       // set the pin as high
    }
                    if((PORTAbits.RA2)&&(PORTAbits.RA3))            // if the pin is high
    {
        //led cyan
                                LED_D7 = 0;
                                LED_D6 = 1;
                                LED_D5 = 1;
        //LATAbits.LATA0 = 1;       // set the pin as high
    }
                    if(PORTAbits.RA1)           // if the pin is high
    {                                                               //led rojo
                                LED_D7 = 1;
                                LED_D6 = 0;
                                LED_D5 = 0;
    }
                    if(PORTAbits.RA2)           // if the pin is high
    {                                                               //led verde
                                LED_D7 = 0;
                                LED_D6 = 1;
                                LED_D5 = 0;
    }
                    if(PORTAbits.RA3)           // if the pin is high
    {                                                               //led azult
                                LED_D7 = 0;
                                LED_D6 = 0;
                                LED_D5 = 1;
    }
    else                    // if the pin is low
    {                                                               //led apagado
                                LED_D7 = 0;
                                LED_D6 = 0;
                                LED_D5 = 0;
    }
                    if  ((PORTAbits.RA4)  ||  (PORTAbits.RA5)  ||  (PORTEbits.RE0)  ||  (PORTEbits.RE1)  ||
(PORTEbits.RE2) ||
                                (PORTCbits.RC0)  ||  (PORTCbits.RC1)  ||  (PORTCbits.RC2)  ||  (PORTDbits.RD0)  ||
(PORTDbits.RD1) ||
                                (PORTDbits.RD2)  ||  (PORTDbits.RD3)  ||  (PORTCbits.RC4)  ||  (PORTBbits.RB3)  ||
(PORTBbits.RB4) ||
                                (PORTBbits.RB5) || (PORTBbits.RB6) || (PORTBbits.RB7))
                    {
                                break; //pulsando cualquier botón, sale del bucle y sigue con su funcion
                    }
   }
            return;

}

//////////////////////////////////////////////////////////////////////////////
void init_ports(void) {

   // As the LED pin is also an analog input pin we have to declare it as
              // digital by writing the port configuration control bits(PCFG<3:0>) in
              // ADCON1 Register - Refer to the datasheet for more details

            ADCON1 = 1111; //configura los pines como digitales, excepto el AN0, AN1, AN2 y AN3 que seguirá como
analógico. (Buscar en el doc "caracteristicas del pic18f4550")
            //ADCON1 = 0x0B; //debería ser lo mismo pero con formato hexadecimal.
```

```
	//Set PORTD bit 0 as digital output, the LED is connected on this pin
	// Si TRISXbits.TRISXn = 1 --> significa que el susodicho pin es de entrada (input) digital

	TRISAbits.TRISA0 = 1; //**se configurará como entrada analógica para el micrófono**//
	TRISAbits.TRISA1 = 1; //DMX Red intput
	TRISAbits.TRISA2 = 1; //DMX Green input
	TRISAbits.TRISA3 = 1; //DMX Blue input
	TRISAbits.TRISA4 = 1; //input RF 4
	TRISAbits.TRISA5 = 1; //input RF 5
	TRISBbits.TRISB0 = 0; //output
	TRISBbits.TRISB1 = 0; //output
	TRISBbits.TRISB2 = 0; //output
	TRISBbits.TRISB3 = 1; //input SW 5
	TRISBbits.TRISB4 = 1; //input SW 6
	TRISBbits.TRISB5 = 1; //input SW 7
	TRISBbits.TRISB6 = 1; //input SW 8
	TRISBbits.TRISB7 = 1; //input SW 9
	TRISCbits.TRISC0 = 1; //input RF 9
	TRISCbits.TRISC1 = 1; //input SW 1
	TRISCbits.TRISC2 = 1; //input SW 2
//	TRISCbits.TRISC4 = 0; //me da error
//	TRISCbits.TRISC5 = 0; //me da error
	TRISCbits.TRISC6 = 1; //input RF 1
	TRISCbits.TRISC7 = 0; //output
	TRISDbits.TRISD0 = 1; //input SW 3
	TRISDbits.TRISD1 = 1; //input SW 4
	TRISDbits.TRISD2 = 1; //input RF 3
	TRISDbits.TRISD3 = 1; //input RF 2
	TRISDbits.TRISD4 = 0; //output
	TRISDbits.TRISD5 = 0; //output LED B
	TRISDbits.TRISD6 = 0; //output LED G
	TRISDbits.TRISD7 = 0; //output LED R
	TRISEbits.TRISE0 = 1; //input RF 6
	TRISEbits.TRISE1 = 1; //input RF 7
	TRISEbits.TRISE2 = 1; //input RF 8

	PORTA = 0x00; //inicializo todos los puertos a 0
	PORTB = 0x00;
	PORTC = 0x00;
	PORTD = 0x00;
	PORTE = 0x00;
	//TRISB = 0xF0 ;	// PORT B Setting: Set all the pins in port B to Output


	//Cuando escribimos un valor a un pin utilitzamos el registro LAT, cuando leemos un valor de un pin usamos el registro PORT
	//Conectamos a la placa de RF los pins RC0 a RC2, RD0, RD1, RC6, RC7, RB5 Y RB 6 (solo funcionan 6 primeros por el mando).
	//Por cada boton que pulsemos se encendará un LED distinto. Las salidas seran de RD2 a RD7 (sin contar los RCs) y del RB0 al RB2.
	//Si LED = 1 --> LED encendido
}


////////////////////////////////////////////////////////////////////////////////
                                                                    /// MAIN ///
////////////////////////////////////////////////////////////////////////////////

 void main(void)
 {
	init_ports();
	while(1)
	{
		if((PORTCbits.RC4) || (PORTCbits.RC1)) //Apretamos el boton1
		{
			//
			rele_Motor();
		}
		else if ((PORTDbits.RD3) || (PORTCbits.RC2)) //Apretamos el boton2
		{
			//
			leds_Rojos();
		}
		else if ((PORTDbits.RD2) || (PORTDbits.RD0)) //Apretamos el boton3
		{
```

```
                        //
                        leds_Verdes();
                }
                else if ((PORTAbits.RA4) || (PORTDbits.RD1)) //Apretamos el boton4
                {
                        //
                        leds_Azules();
                }
                else if ((PORTAbits.RA5) || (PORTBbits.RB3)) //Apretamos el boton5
                {
                        //
                        leds_Amarillos();
                }
                else if ((PORTEbits.RE0) || (PORTBbits.RB4)) //Apretamos el boton6
                {
                        //
                        leds_Magenta();
                }
                else if ((PORTEbits.RE1) || (PORTBbits.RB5)) //Apretamos el boton7
                {
                        //
                        apaga_Leds();
                }
                else if ((PORTEbits.RE2) || (PORTBbits.RB6)) //Apretamos el boton8
                {
                        //
                        Arrancar();
                }
                else if ((PORTCbits.RC0) || (PORTBbits.RB7)) //Apretamos el boton9
                {
                        //
                        Microfono();
                }
                else if ((PORTAbits.RA1) || (PORTAbits.RA3) || (PORTAbits.RA2)) //Recibimos senyal DMX
                {
                        //
                        leds_Dmx();
                }
                else
                {
                        Arrancar();
                }
        }
}

////////////////////////////////////////////////////////////////////////////////////////////////////
```

ANNEX II. CODIFICACIÓ ARDUINO UNO

```
/*****************************************************************************************************************
*
* Title                                 : Example DMX Receiver
* Version                 : v0.1
* Last updated                  : 29.04.2012
* Target                  : Arduino mega 2560, Arduino mega 1280, Arduino nano
* Author                : Toni Merino - merino.toni at gmail.com
* Web                   : www.deskontrol.es
*
*****************************************************************************************************************/
#include <lib_dmx.h>  // comment/uncomment #define USE_UARTx in lib_dmx.h as needed

void setup()
{
  ArduinoDmx0.set_control_pin(22);      //max485 input/output control (connect to 485 pins 2-3)
  //ArduinoDmx1.set_control_pin(24);
  //ArduinoDmx2.set_control_pin(26);
  //ArduinoDmx3.set_control_pin(28);

  ArduinoDmx0.set_rx_led_pin(13); //30     //rx signal indicator led pin, don't care
  //ArduinoDmx1.set_rx_led_pin(31);
  //ArduinoDmx2.set_rx_led_pin(32);
  //ArduinoDmx3.set_rx_led_pin(33);

  //ArduinoDmx0.set_tx_led_pin(34);      //tx signal indicator led pin, don't care
  //ArduinoDmx1.set_tx_led_pin(35);
  //ArduinoDmx2.set_tx_led_pin(36);
  //ArduinoDmx3.set_tx_led_pin(37);

  cli();          //disable interrupts

  ArduinoDmx0.set_rx_address(32);      //set rx0 dmx start address
  ArduinoDmx0.set_rx_channels(3);     //number of rx channels
  ArduinoDmx0.init_rx(0);               //starts universe 0 as rx

  sei();          //enable interrupts
}  //end setup()

void loop()
{
  for(;;)
  {
    //write values from dmx channels 1-4 universe 0 to arduino pwm pins 5-8
    analogWrite(5, ArduinoDmx0.RxBuffer[0]);  // led Rojo -- buffers 0 indexed
    analogWrite(6, ArduinoDmx0.RxBuffer[1]);  //led Verde
    analogWrite(7, ArduinoDmx0.RxBuffer[2]);  //led Azul
    //analogWrite(8, ArduinoDmx0.RxBuffer[3]);
  }
}  //end loop()
```